Optimización mediante algoritmo de hormigas aplicado a la recolección de residuos sólidos en UNAM-CU

Elizabeth Mancera-Galván, Beatriz A. Garro-Licón, Katya Rodríguez-Vázquez

IIMAS-UNAM, Ciudad Universitaria, D.F.,
México

elizabethal_20@hotmail.com,
{beatriz.garro, katya.rodriguez}@iimas.unam.mx

Resumen. En este artículo se aplica la metaheurística de colonia de hormigas (ACO) para resolver el problema de ruteo que se presenta al realizar la tarea de recolección de residuos sólidos en UNAM-CU. Este espacio está dividido en varios circuitos de los cuales el circuito CCU será nuestro primer caso de estudio. El encontrar la mejor ruta para este circuito, puede verse como un problema de optimización, que en una primera etapa se plantea como el problema clásico del agente viajero (TSP). Para resolver el problema, cuatro algoritmos ACO son utilizados: Sistema de hormigas (AS), Sistema de hormigas elitista (EAS), Sistema de hormigas Max-Min (MMAS) y Sistema de colonia de hormigas (ACS). Los resultados muestran reducción en la distancia recorrida con respecto a la ruta actualmente adoptada empíricamente en CU así como el desempeño de los algoritmos al realizar un estudio previo de sensibilidad de los parámetros.

Palabras clave: optimización por colonia de hormigas, ruteo de vehículos, TSP, recolección de residuos sólidos.

1. Introducción

Actualmente, la aplicación de modelos matemáticos a problemas reales de ruteo ha cobrado gran importancia debido al impacto positivo reflejado en costos, tiempo y calidad de servicio. Dentro de estos problemas reales, podemos encontrar la optimización de sistemas de recolección de residuos en zonas urbanas [10, 14, 15], cuyos objetivos principales son la reducción del uso de vehículos para el transporte, minimizar la distancia recorrida y cubrir un servicio a un cliente, donde se maximicen ganancias y el tiempo empleado sea mínimo.

La recolección de residuos, puede efectuarse por diferentes métodos, entre los que destacan por contenedores o aceras. El primer caso se plantea como un problema de ruteo por nodos (VRP). Mientras que el segundo trata el problema como ruteo de arcos (CARP). En el caso del VRP con un solo vehículo el problema se reduce al clásico problema del agente viajero (TSP) [2, 8] y en el caso del CARP, este puede ser planteado como el problema del cartero chino [6]. Este último, considera un carte-

ro cuya tarea es entregar la correspondencia en un vecindario. Para lograrlo, es necesario que parta de la oficina de correros, recorra todas las calles (arcos) y regrese a dicha oficina, de tal manera que camine la menor distancia posible sin repetir arcos y con la posibilidad de pasar por un mismo nodo varias veces (intersecciones entre arcos).

Para resolver el problema de recolección de residuos, se han utilizado métodos exactos [7, 12, 13], no obstante, la aplicación de estos métodos es limitada pues problemas complejos como el VRP y el TSP, pueden volverse NP-duros. En este caso, los métodos exactos ya no encuentran una solución óptima en un tiempo razonable. Es por esto, que se acude a la aplicación de técnicas más eficientes como las heurísticas, que si bien, no siempre encuentran las soluciones óptimas, son capaces de resolver el problema en un tiempo polinomial y de no alcanzar el óptimo, las soluciones encontradas son buenas aproximaciones al mismo.

En la literatura, el problema de recolección de residuos ha sido estudiado y abordado desde las dos perspectivas mencionadas anteriormente (VRP y CARP). Algunos trabajos publicados que pueden ser citados son: el de Thierry Kulcar [12], en cuyo trabajo se representa el problema de recolección de residuos visto como un CARP. El objetivo de este problema es minimizar los costos de transporte de residuos en Bruselas, al utilizar un método de ramificación y acotamiento. Por otra parte, Bonomo *et al.* [1] presentan un modelo de TSP para diseñar las rutas de recolección en el sur de Buenos Aires. Hornig y Fuentealba [8], aplican un algoritmo ACO a la recolección de residuos por contenedores, en un municipio de Chile. Por último, Karadimas *et al.* [11], también aplican un algoritmo ACO para establecer rutas de recolección en una zona de Grecia.

En este trabajo, se utiliza la metaheurística ACO para resolver un problema de recolección de residuos sólidos, visto como un TSP; donde el objetivo, es minimizar la distancia recorrida por los vehículos. Nuestro caso de estudio se enfoca al ruteo de vehículos al recolectar residuos sólidos en UNAM-CU (Universidad Nacional Autónoma de México-Ciudad Universitaria). Este problema en la actualidad, está resuelto de manera empírica, lo cual impide que la recolección de residuos se realice de manera eficiente en tiempo y costo para la institución. Este trabajo resulta importante debido a la aplicación a un problema real, en donde no han sido utilizadas técnicas de optimización para resolver un problema de ruteo tan importante en la UNAM-CU. Cabe destacar que un problema de recolección de residuos mal planeado genera problemas de tránsito, de insalubridad, elevados costos monetarios y de tiempo. Por este motivo, se propuso investigar sobre el tema y generar una solución a la planificación de dicho ruteo en la UNAM-CU.

Este artículo se encuentra organizado de la siguiente manera: en la Sección 2 se presenta a detalle el caso de estudio a resolver, mientras que en la Sección 3 se presentan los diferentes algoritmos ACO que serán aplicados al TSP. En la Sección 4, se presentan los resultados obtenidos al realizar un análisis de sensibilidad de los parámetros de los algoritmos y la aplicación de los mismos al resolver el problema dado. Finalmente, las conclusiones y trabajos futuros son descritos en la Sección 5.

2. Descripción del problema y formulación matemática

Ciudad Universitaria (CU), uno de los centros educativos más importantes de México, se localiza en el sur de la ciudad de México y cuenta con 730 hectáreas de superficie. La Dirección General de Conservación y Obras (DGOC) de la UNAM calculó que en promedio en CU, se generan 15 toneladas de basura con una población que es superior a las 150,000 personas. Esto sugiere que en promedio se generan diarios 0.1 Kg de residuos por persona.

El problema de recolección de residuos sólidos en CU, es el principal caso de estudio en este trabajo debido a la necesidad diaria de mantener en condiciones salubres la Universidad y reducir los costos que esto genera. Este caso, puede ser resuelto si se considera como un problema de optimización combinatoria, donde se busca obtener la mínima distancia recorrida en la ruta que abarca un número considerable de puntos de recolección y rutas a seguir por los vehículos.

Por políticas previamente establecidas, la zona de CU se encuentra dividida en dos áreas (CCU y ZN), en cada una de ellas existe un depósito donde los camiones inician y finalizan su ruta. Para propósitos de este artículo, se trabajará solamente con el área CCU. Sin embargo, en un próximo trabajo se pretende incorporar el resto de la zona, para ser resuelto como un problema asimétrico de VRP, utilizando más de un vehículo con capacidades limitadas y con otras restricciones.

En CU la Dirección General de Obras y Conservación, un organismo interno propio de la universidad, es el encargado de coordinar la recolección de residuos sólidos. Los datos sobre la localización de los puntos de recolección, así como la ruta y el mapa de la zona que más adelante se muestran, fueron obtenidos de dicho organismo [9].

Actualmente, en CCU operan con un solo vehículo y en total se localizan 31 puntos de recolección que se encuentran distribuidos como se muestra en la Figura 1. Los puntos de recolección 25, 26, 27 y 28 han sido agrupados en uno solo, pues los residuos de los puntos 26, 27 y 28 se concentran en el 25, previo a la recolección. La distancia recorrida diariamente por el vehículo de la DGOC en esta zona es de 34,132.37 m, siguiendo la numeración consecutiva de los diferentes puntos.

El camión recolector trabaja dos turnos: en la mañana y en la tarde; y cada recorrido se lleva a cabo siguiendo el orden de los puntos de recolección. Cabe mencionar, que antes de regresar al depósito, el camión debe ir a dejar los residuos a un depósito del gobierno del Distrito Federal, conocido como "Estación de transferencia". Esta estación se ubica lejos de los límites de CU, pero en el mapa queda representada cerca de las inmediaciones de CU sólo para su visualización.

En este trabajo, el problema de recolección de residuos se plantea como un TSP, en donde el vehículo partirá del depósito, recorrerá todos los puntos de recolección sólo una vez y regresará al depósito después de haber ido a la estación de transferencia.

Formalmente, el problema puede ser representado por un grafo completo, dirigido y asimétrico G=(V,A), donde V={0,1,…,n+1} es el conjunto de vértices (puntos de recolección) y A es el conjunto de arcos (trayectorias entre puntos de recolección). El vértice D={0} representa el depósito, los vértices V'={1,2,..,n} representan los puntos de recolección, y el vértice I={n+1}, representa la estación de transferencia. Por otro

lado, a cada arco se le asocia un costo c_{ij} no negativo, el cual representa la distancia entre los puntos de recolección *i* y *j*.

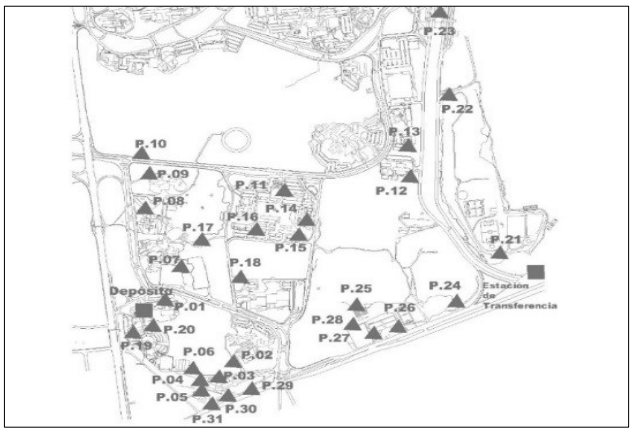

Fig. 1. Puntos de recolección en CCU.

El modelo matemático para nuestro problema de ruteo, el cual se basa en el clásico TSP, se presenta a continuación. Donde x_{ij} es una variable binaria que toma el valor de uno si el arco (i, j) se encuentra en el tour óptimo y cero en otro caso.

$$Min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{1}$$

Sujeto a

$$\sum_{\substack{i \in V \\ i \neq j}} x_{ij} = 1 \qquad \forall j \in V \tag{2}$$

$$\sum_{\substack{j \in V \\ j \neq \mathrm{i}}} x_{ij} = 1 \qquad \forall i \in V \tag{3}$$

$$x_{n+1,0} = 1 \tag{4}$$

$$x_{i0} = 0 \qquad \forall i \in V' \tag{5}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \quad \forall S \subseteq V \backslash \{0\}, |S| \geq 2 \tag{6}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in V \tag{7}$$

La Ecuación (1) representa la función objetivo, la cual consiste en la minimización de la distancia total de la suma de los arcos utilizados en el tour. La restricción (2) señala que todos los puntos de recolección deben ser visitados exactamente una vez, mientras que la restricción (3) indica que el vehículo debe salir del punto que visitó. En la restricción (4) se señala que después de que el vehículo haya ido a la estación de transferencia, sólo podrá dirigirse al depósito. La restricción (5) expresa que el vehículo no puede regresar al depósito después de haber visitado un punto de recolección. La restricción (6) asegura la continuidad del tour, es decir, que el tour sea conexo y así evitar la generación de pequeños sub-tours. En (7) se especifica que la variable x_{ij} sólo puede tomar los valores uno o cero.

Una vez identificado el problema que queremos resolver, procedemos a explicar en la siguiente Sección, qué algoritmos se utilizan para dar solución al problema de recolección de residuos en CCU.

3. Algoritmos ACO

Los algoritmos basados en la optimización por colonia de hormigas (Ant Colony Optimization-ACO), se encuentran dentro de las metaheurísticas denominadas como Inteligencia de Enjambres. Este tipo de metaheurísticas, simulan el comportamiento social de enjambres de insectos para resolver problemas de optimización. La metaheurística ACO simula el comportamiento de las hormigas reales cuando éstas se encuentran buscando comida. Específicamente, las hormigas son capaces de encontrar el camino más corto entre el nido y la fuente de alimento mediante un tipo de comunicación indirecta, la cual se basa en seguir un rastro de feromona que es depositado por cada hormiga a su paso y reforzado a su regreso al nido [2]. La simulación en ACO sobre el comportamiento de las hormigas reales es mediante el uso de hormigas artificiales, las cuales son capaces de aprender sobre el espacio de búsqueda durante la ejecución del algoritmo y usan esta experiencia adquirida para construir, mejores soluciones en cada iteración. Este proceso de construcción puede entenderse como una toma secuencial de decisiones regida por una regla de transición estocástica.

Parte esencial de los algoritmos ACO, es la combinación de información heurística y el rastro de feromona. La información heurística, también llamada *visibilidad*, mide lo deseable que es un nodo *j* para ser visitado desde *i*, con base en la información a priori del problema. Vale la pena mencionar que, esta información no es modificada por las hormigas. En cuanto al rastro de feromona, intuitivamente se puede decir que mide qué tan deseable es un nodo con base en lo que han aprendido las hormigas. En este caso, el rastro de feromona es modificado por dichos insectos virtuales.

El parámetro τ refleja el rastro de feromona, mientras que η representa la información heurística (visibilidad). Ambos parámetros son utilizados en el algoritmo para calcular la probabilidad que permitirá a cada hormiga decidir cómo moverse a través del grafo.

La metaheurística ACO fue introducida por Marco Dorigo [5] y a partir de dicho desarrollo se han generado diversas propuestas que buscan mejorar el funcionamiento del algoritmo original. A continuación, se describe el algoritmo básico llamado Sistema de hormigas (Ant System-AS) y tres de sus variantes: Sistema de hormigas elitista (Elitist Ant System-EAS), Sistema de hormigas Max-Min (Max-Min Ant system-MMAS) y Sistema de colonia de hormigas (Ant Colony System-ACS).

3.1 Sistema de hormigas (AS)

El primer algoritmo propuesto dentro de la metaheurística ACO fue el Sistema de hormigas [4, 5]. De manera general, en el algoritmo AS, se posiciona un número de hormigas k en cada nodo. Cada hormiga construye una solución factible al problema, al aplicar de manera iterada la siguiente regla de transición que combina τ y η para elegir con cierta probabilidad p_{ij} la siguiente ciudad *j* a visitar desde la ciudad *i*.

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{u\epsilon \mathrm{N}_i^k} \tau_{iu}^{\alpha}\eta_{iu}^{\beta}} & si\ \ j\epsilon \mathrm{N}_i^k \\ 0 & en\ otro\ caso \end{cases} \quad (\mathbf{8})$$

Donde los parámetros α y β, determinan la importancia de la feromona y la información heurística, respectivamente. Por otro lado η_{ij}, es el recíproco de la distancia entre *i* y *j*.

Una vez que todas las hormigas han construido un tour completo, actualizan la feromona, de acuerdo a la Ecuación (9).

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \quad (\mathbf{9})$$

Donde ρ es la tasa de evaporación, *m* es el número de hormigas, y $\Delta\tau_{ij}^{k}$ es la cantidad de feromona depositada en el arco (*i,j*) por la hormiga *k*. $\Delta\tau_{ij}^{k}$ y se calcula como Q/L_k, con L_k que representa la longitud del tour construido si (*i,j*) pertenece al tour hecho por la hormiga *k* y es cero en otro caso.

3.2 Sistema de hormigas elitista (EAS)

Esta propuesta es una variante del algoritmo original AS [2], en la que se busca dar un peso adicional a la mejor solución conocida hasta el momento s_{bs} (mejor solución global) en la actualización de la feromona. Esta solución es generada por una hormiga, a la cual se le llama *hormiga elitista*. Para construir las soluciones en EAS se sigue la Ecuación (8) como en el algoritmo AS. Para actualizar la feromona, se agrega una cantidad ($\Delta\tau_{ij}^{bs} = \frac{1}{L_{best}}$) de feromona extra en los arcos que pertenecen a la mejor solución global generada por la hormiga elitista. La actualización de feromona modificada es la siguiente:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{s_{bs}} \quad (\mathbf{10})$$

donde e es un parámetro que pondera la influencia de la mejor solución global.

3.3 Sistema de hormigas Max-Min (MMAS)

El algoritmo sistema de hormigas Max-Min [16] toma como base el algoritmo AS, pues utiliza la misma regla de construcción de soluciones (Ecuación (8)). Sin embargo, se añaden tres modificaciones al algoritmo básico, las cuales se generan tomando en cuenta que en el MMAS se busca explotar la mejor solución global o la mejor solución encontrada en la iteración actual. Las modificaciones son: el rastro de feromona se encuentra acotado inferior y superiormente; el valor inicial de feromona queda definido como el límite superior del mismo y por último, los niveles de feromona son reinicializados si durante cierto número de iteraciones, la solución no mejora. La actualización de feromona queda definida por la Ecuación (11).

$$\tau_{ij} = \left[(1-\rho)\tau_{ij} + \Delta\tau_{ij}^{s_{bs}}\right]_{\tau_{min}}^{\tau_{max}} \quad (\mathbf{11})$$

donde τ_{max} y τ_{min}, son el límite superior e inferior de los niveles de feromona respectivamente. Para calcularlos, usamos las ecuaciones presentadas en [16].

3.4 Sistema de colonia de hormigas (ACS)

El algoritmo sistema de colonia de hormigas [3] es la variante que más difiere del AS, al integrar tres cambios. En primer lugar, para construir soluciones las hormigas utilizan la Ecuación (12). Donde *j* es el siguiente nodo a visitar, q es una variable aleatoria uniforme en [0,1], q_0 es un parámetro entre [0,1]. J denota el nodo a elegir siguiendo la Ecuación (8).

$$\mathrm{j} = \begin{cases} \arg max_{\mathrm{l}\epsilon \mathrm{N}_i^k} \left({\tau_{ij}}^{\alpha} \eta_{ij}^{\beta}\right) & si \;\; q \leq q_0 \\ J & en\ otro\ caso \end{cases} \quad (\mathbf{12})$$

Como segunda y tercera modificación, en el algoritmo ACS, la actualización de feromona queda dividida en dos procesos: uno global y uno local. En la actualización local todas las hormigas modifican el nivel de feromona cada vez que atraviesan un arco. Esta regla (Ecuación (13)), conocida como regla de actualización local de feromona, está determinada por:

$$\tau_{ij} = (1-\varphi)\tau_{ij} + \varphi\tau_0 \quad (\mathbf{13})$$

donde φ es un coeficiente de decaimiento que toma valores en (0,1], y τ_0 es el valor inicial de feromona.

Para realizar la actualización global, se toma en cuenta una sola hormiga, la que generó la mejor solución global. La Ecuación (14) es utilizada para añadir la feromona a los arcos, después de cada iteración.

$$\tau_{ij} = \begin{cases} (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{best} & si\ \ (i,j)\ está\ en\ el\ mejor\ tour \\ \tau_{ij} & en\ otro\ caso \end{cases} \quad (14)$$

4. Resultados experimentales

Diversos experimentos se llevaron a cabo para validar la solución propuesta utilizando las distancias entre todos los posibles puntos de recolección.

Para definir los valores de los parámetros utilizados en los algoritmos ACO, se tomaron como base los siguientes: para el algoritmo de AS los parámetros fueron *m*=10, α=1, β=3, $\tau_o = 1/L_{nn}$[1], ρ=0.1, Q=1, para el EAS además de los anteriores se incluye *e*=1. Por otra parte, para el algoritmo ACS, se utilizan también los parámetros φ =0.1 y $q_0 = 0.9$. Por último, en el MMAS el mecanismo de reinicialización es activado si después de 250 iteraciones la solución no mejora. Estos parámetros fueron elegidos porque son de los más utilizados en la literatura y aunque en ACS estos valores varían, a partir de los mismos parámetros es posible comparar el comportamiento de los algoritmos. Como se verá en la discusión de resultados, fue necesario evaluar la influencia de cada parámetro tomando diferentes valores para cada uno dentro de sus rangos establecidos.

El número de iteraciones realizadas fueron 1000 para todos los algoritmos y se realizaron 30 corridas para cada uno. Dado que los algoritmos ACO son heurísticos y dependen de los valores iniciales para realizar su búsqueda de soluciones, los resultados pueden otorgar valores diferentes en cada una de las corridas del algoritmo. Por lo que, para validar el resultado de los mismos es necesario realizar vari0s experimentos.

En la Tabla 1, se muestran los resultados obtenidos con los parámetros ya especificados anteriormente. Se puede observar que los cuatro algoritmos llegan a la misma solución (31742 mts). Esta distancia resulta menor al ser comparada con la distancia recorrida actualmente (generada empíricamente) en CCU por vehículos de la DGOC, la cual es de 34132.37 mts. Por otro lado, se aplicó la heurística del vecino más cercano para comparar los resultados obtenidos con los algoritmos ACO. La distancia encontrada por esta heurística fue de 35746.3 mts.

A pesar de que todos los algoritmos arrojan la misma solución, el MMAS presenta una mayor estabilidad. Esto puede observase en el promedio de las soluciones ya que para el MMAS el valor es menor entre los cuatro algoritmos así como la desviación estándar obtenidas en las 30 corridas. Adicionalmente, el número de iteraciones promedio para encontrar la mejor solución resulta menor comparado con el AS, EAS y ACS. Otro factor que influye para determinar que la estabilidad del algoritmo MMAS es la desviación estándar del número de iteraciones ya que es menor. Esto se traduce en que, para el MMAS se necesitan entre 341 y 602 iteraciones para encontrar el óp-

[1] L_{nn} es la longitud del tour mínimo encontrado utilizando la técnica de vecino más cercano.

timo, mientras que para los tres algoritmos restantes, la solución óptima podría encontrarse después de las 800 iteraciones.

Tabla 1. Resultados.

Parámetro	Algoritmo	Mejor solución (B)	Promedio soluciones (P)	Peor Solución (W)	Desv. Estándar (S)	Prom. Iter. (PI)	Desv. Estándar Iter. (SI)
Base	AS	31742.0	32179.0	32822.1	393.0	537.7	310.7
	EAS	31742.0	31905.1	33129.2	308.4	510.6	330.3
	MMAS	31742.0	31899.7	32904.8	202.6	471.7	130.3
	ACS	31742.0	32340.4	33471.0	602.0	591.7	233.8
β=0.5	AS	32380.6	33340.3	34196.5	469.4	671.8	207.5
	EAS	31742.0	32090.0	32865.0	343.6	727.0	252.0
	MMAS	31742.0	32616.8	33658.4	444.7	684.9	198.9
	ACS	31912.7	33435.7	34848.0	667.8	138.7	25.6
β=1	AS	31742.0	32256.5	32683.8	348.7	456.2	278.5
	EAS	31742.0	31814.1	32604.0	172.2	507.4	226.1
	MMAS	31742.0	31805.4	32394.8	150.3	597.0	144.8
	ACS	31742.0	32202.7	33043.9	481.3	458.2	276.4
β=5	AS	31742.0	32738.9	33416.0	391.7	348.4	322.2
	EAS	31742.0	32311.7	33554.3	586.0	543.6	332.6
	MMAS	31742.0	31997.0	33129.2	297.8	569.9	215.2
	ACS	317420	32548.6	33605.1	439.8	600.4	248.9
β=7	AS	32474.3	33089.8	34152.4	447.0	312.7	281.4
	EAS	31742.0	32955.5	34447.7	576.8	432.6	274.0
	MMAS	30993.5	31837.7	32552.3	377.1	534.5	237.5
	ACS	31762.0	32971.7	34200.0	669.5	261.8	200.3
m=5	AS	31880.3	32570.5	33264.7	344.8	612.4	267.6
	EAS	31742.0	32471.7	33947.5	592.7	549.4	336.3
	MMAS	31742.0	32716.7	34150.7	884.8	518.4	315.1
	ACS	31880.3	32833.8	34208.9	601.5	133.0	70.9
m=20	AS	317420	32004.2	32602.1	236.0	392.8	300.7
	EAS	31742.0	31849.4	31900.3	58.8	367.1	251.2
	MMAS	31742.0	31802.8	31968.9	73.0	474.7	113.6
	ACS	31742.0	32175.4	33605.1	588.1	403.8	255.5
m=25	AS	31880.3	32003.7	32602.1	245.6	391.4	298.5
	EAS	31742.0	31848.0	31880.3	59.5	311.4	270.9
	MMAS	31742.0	31909.3	33146.2	350.1	572.8	257.9
	ACS	31742.0	32176.8	33159.9	498.8	84.9	18.5
m=29	AS	31742.0	32011.9	32632.6	248.0	285.6	275.9
	EAS	31742.0	31844.1	31900.3	62.7	389.0	295.8
	MMAS	31742.0	31756.3	31880.3	38.8	446.4	86.4
	ACS	31742.0	31932.4	32990.9	338.2	570.8	321.2

Parámetro	Algo-ritmo	Mejor solución (B)	Promedio soluciones (P)	Peor Solución (W)	Desv. Estándar (S)	Prom. Iter. (PI)	Desv. Estándar Iter. (SI)
ρ=0.01	AS	31742.0	32095.2	32778.4	306.6	500.4	289.1
	EAS	31742.0	31918.1	32802.9	252.7	511.0	276.7
	MMAS	31742.0	32494.0	33397.5	512.3	112.2	103.2
	ACS	31742.0	32114.6	32891.8	353.0	535.2	287.9
ρ=0.3	AS	31762.0	32427.5	33129.2	389.2	411.9	355.3
	EAS	31742.0	32103.2	33004.3	411.5	476.9	270.0
	MMAS	32612.5	34116.4	36825.3	949.6	30.3	11.6
	ACS	31742.0	32173.1	33872.8	597.4	432.1	254.6
ρ=0.5	AS	31880.3	32552.0	32961.4	310.3	391.3	287.0
	EAS	31742.0	32419.9	33265.3	504.1	365.0	284.6
	MMAS	31742.0	32306.8	33972.0	700.3	580.7	296.3
	ACS	33212.7	35284.5	38193.9	1323.3	18.3	8.7
ρ=0.7	AS	31742.0	32728.3	33221.1	392.5	412.4	307.4
	EAS	31742.0	32342.3	32994.4	480.2	565.8	313.8
	MMAS	31742.0	32403.6	33783.9	652.3	450.4	321.7
	ACS	32691.5	36406.6	40255.8	1822.6	10.6	4.8
e=3	EAS	31742.0	32135.9	33649.5	473.7	297.5	240.2
e=5	EAS	31742.0	32097.1	33254.0	493.7	403.9	287.3
e=7	EAS	31742.0	32399.9	33397.5	561.7	398.2	306.1
e=10	EAS	31742.0	32597.7	34057.6	586.4	380.5	333.8
Sin reini-cialización	MMAS	31742.0	31969.3	32990.9	276.2	521.6	163.8
S^{bs}	MMAS	31742.0	32530.1	34031.1	684.4	280.8	95.8
φ=0.01	ACS	31742.0	32428.0	34529.2	876.5	539.4	308.5
φ=0.3	ACS	31742.0	31902.4	32474.3	248.8	663.8	229.2
φ=0.5	ACS	31742.0	32092.9	33383.6	427.5	632.9	258.9
φ=0.7	ACS	31742.0	32151.5	33272.0	383.0	595.9	238.3
q_0=0.3	ACS	31742.0	31991.7	33405.7	478.4	710.8	258.4
q_0=0.5	ACS	31742.0	32158.4	33146.2	541.4	574.7	242.8
q_0=0.7	ACS	31742.0	32070.6	33129.2	446.6	557.7	217.2
q_0=0.95	ACS	31742.0	32177.5	33720.3	624.5	574.5	266.2

4.1 Discusión de los resultados

Una forma de entender mejor el comportamiento de los algoritmos ACO, es mediante el análisis de sensibilidad aplicado a los parámetros de dichos algoritmos. En este trabajo los parámetros tomados en cuenta para llevar a cabo el análisis de sensibilidad, son los siguientes: m, β, ρ, e, φ y q_0, según el algoritmo. Es importante mencionar que si uno de los parámetros es modificado, los demás permanecen constantes, con los valores definidos previamente. Todo el análisis se basa en los resultados mostrados en la Tabla 1.

Dado que casi siempre se llega a la solución óptima, nos centraremos en revisar el promedio y desviación estándar de las soluciones e iteraciones. La razón de tal revisión es que si bien, con todos los algoritmos encontramos el óptimo en este problema, es importante comparar el tiempo que tardan en converger a la solución. Esto se puede analizar al observar el número de iteraciones que se necesitan para alcanzar los mejores resultados. De manera similar, es deseable que en las 30 corridas, las soluciones obtenidas no difieran en gran cantidad, lo cual se puede observar con el promedio y desviación estándar de las soluciones.

De acuerdo a β en todos los algoritmos, a excepción del AS, existe una mejora del promedio de las soluciones (P) cuando β=3 (ver Figura 2), y empeoran cuando β aumenta o disminuye; comportamiento que también presenta la desviación estándar del promedio (S). Cuando β=7, los algoritmos MMAS, AS y ACS no encuentran la mejor solución (31742.0). Por otro lado, aunque el promedio de iteraciones (PI) no siempre es menor cuando β=3; al comparar con los demás valores, se observa que conforme disminuye dicho promedio, el promedio de las soluciones aumenta. Dado lo anterior, podemos decir que para los algoritmos EAS, MMAS y ACS un valor de β=3, permite que las hormigas encuentren mejores soluciones.

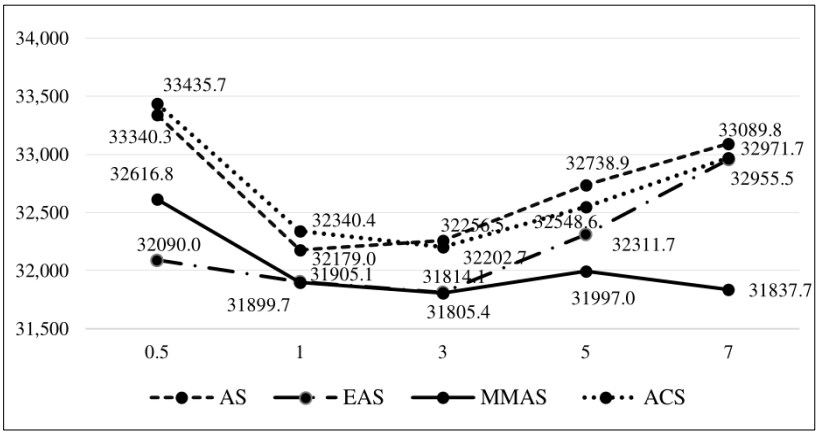

Fig. 2. Análisis del promedio de soluciones para β.

De manera general, para el parámetro *m* (número de hormigas), utilizar más hormigas genera cambios favorables en el promedio de las soluciones (ver Figura 3). Si se utilizan 29 hormigas (igual al número de puntos de recolección) P disminuye en todos los algoritmos, con respecto a los resultados obtenidos cuando *m*=10. Este hecho tiene mayor impacto en el algoritmo ACS. Por otra parte, cuando se utilizan 20 o 25 hormigas, las diferencias con respecto a utilizar 29 son muy pequeñas, excepto para ACS. Sin embargo, el PI para ACS y EAS aumenta si *m*=29. Podemos concluir, para este trabajo, que utilizar un número cercano al número de puntos de recolección mejora las soluciones obtenidas y, en el caso del MMAS y AS, permite que los algoritmos tengan una rápida convergencia.

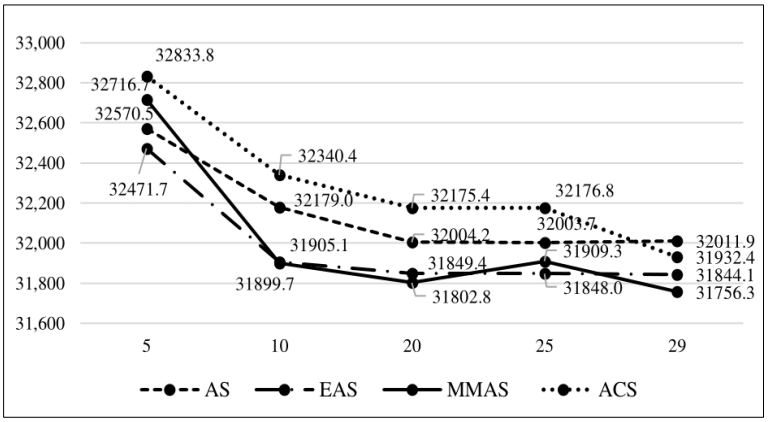

Fig. 3. Análisis del promedio de soluciones para *m*.

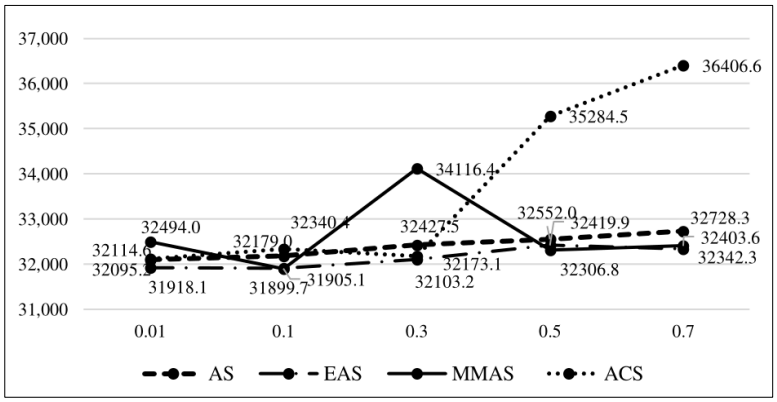

Fig. 4. Análisis del promedio de soluciones para ρ.

Si comparamos el comportamiento de los cuatro algoritmos, al modificar los valores de ρ, es interesante que para EAS y MMAS aumentar o disminuir el valor de dicho parámetro repercute negativamente en P en especial para el algoritmo Max-Min (ver Figura 4). Otra cuestión importante, se observa en PI, pues para el algoritmo MMAS el promedio de iteraciones disminuye considerablemente cuando ρ=0.01 o ρ=0.3, es decir, el algoritmo se estanca rápidamente en una mala solución sin encontrar la mejor. Dado lo anterior, podemos decir que el valor de ρ=0.1, genera las mejores soluciones para EAS y MMAS. Por otra parte, al disminuir el valor de ρ a 0.01, AS y ACS mejoran el promedio de las soluciones al igual que el tiempo de convergencia a la solución. Sin embargo, al aumentar el valor de dicho parámetro, el promedio de las soluciones empeora, comportamiento que también presenta la desviación estándar del promedio (S). Más aun, con tales valores, no se encuentra la mejor solución. Al observar la convergencia del ACS, se puede inferir que el algoritmo se estanca rápidamente en una mala solución. Por lo tanto, para AS y ACS, parece ser más conveniente utilizar un parámetro ρ=0.01.

De manera particular, para el parámetro *e* (solamente utilizado en EAS), se observa que el número de iteraciones promedio disminuye al aumentar dicho valor. Sin embargo el promedio de las soluciones empeora de manera significativa. Además, la desviación estándar del promedio (S) también aumenta conforme el valor de *e* lo hace; por lo tanto, utilizar un valor de *e*=1, otorga los mejores resultados.

Enfocándonos en el mecanismo de reinicialización del MMAS, se observa que al quitarlo, el promedio de las soluciones no mejora, incluso el promedio de iteraciones aumenta. Por otro lado, utilizar la mejor solución global en lugar de la mejor solución de la iteración, tiene un efecto negativo en el promedio de soluciones y, solamente hace que el algoritmo converja más rápido. Como es mencionado en [16], para problemas con una dimensión no muy grande, es preferible utilizar la mejor solución en la iteración para actualizar la feromona. Por lo tanto, para nuestro problema utilizar el mecanismo de reinicialización, así como actualizar la cantidad de feromona mediante la mejor solución de la iteración, arroja mejores resultados con el MMAS.

Particularmente en el algoritmo ACS se utiliza el parámetro φ, que funciona como un reductor de feromona. Para nuestro trabajo, se puede observar que cuando disminuye a 0.01, el promedio de las soluciones empeora, pero si aumenta, el promedio de soluciones mejora, obteniéndose el mejor resultado al fijar el valor en 0.3. También se puede observar una relación entre el promedio de las soluciones y el promedio de iteraciones, ya que conforme el primero mejora, el segundo aumenta. Esto significa que al aumentar el valor de este parámetro a 0.3 permitimos que haya mayor exploración, es decir, ampliar la búsqueda en el espacio de soluciones lo que implica una convergencia tardía. Sin embargo, si φ rebasa el valor de 0.5, ya no se genera una mejora en el promedio de las soluciones. Dado lo anterior, se concluye que utilizar un valor φ=0.3, mejoraría los resultados obtenidos con ACS.

Otro parámetro utilizado específicamente en ACS es q_0, este parámetro permite que haya mayor exploración si este es menor. Dado los resultados que se muestran en la Tabla 1, cuando existe mayor exploración mejora el promedio en las soluciones, obteniendo el mejor resultado con $q_0 = 0.7$. Sin embargo, al utilizar dicho valor la convergencia a la solución es más lenta.

5. Conclusiones

En este artículo se resolvió un problema real de ruteo para la recolección de residuos en UNAM-CCU. Este problema fue planteado como un TSP asimétrico con un par de restricciones extras, necesarias para la adaptación del problema. Para resolverlo, utilizamos cuatro algoritmos ACO, y con todos se encontró la misma solución, lo que se debe a que el número de puntos de recolección no es muy grande. Además, con nuestra propuesta logramos reducir en 7% la distancia total recorrida actualmente de manera empírica en CCU. Los resultados generados con los diferentes algoritmos ACO fueron comparados con el algoritmo de vecino más cercano cuyo resultado indica que se llegó al mínimo recorrido al aplicar heurísticas ACO. Esta reducción resulta importante porque, además de minimizar la distancia, se tiene un efecto positivo en tiempos y costos. Por otra parte, al comparar el promedio de las soluciones obtenidas

durante todas las corridas, los mejores resultados se obtienen para el algoritmo MMAS y los peores para el AS. Aunque con el algoritmo AS las soluciones obtenidas no son tan buenas como las que proporcionan sus sucesores, también se puede mencionar que es el algoritmo ACO más sencillo de implementar y el que involucra un menor número de parámetros. Por esta razón, utilizarlo para problemas pequeños es conveniente.

Con base en el análisis de sensibilidad, podemos mencionar que los parámetros tienen diferente impacto en los resultados obtenidos con los cuatro algoritmos. De manera particular, en nuestro caso de estudio se tiene, por ejemplo, que al aumentar el número de hormigas todos los algoritmos presentan un mejor desempeño. Por otra parte, disminuir la tasa de evaporación impacta positivamente al AS y ACS y, por el contrario, afecta al MMAS y al EAS. Adicionalmente, es interesante observar que el algoritmo MMAS y EAS obtienen soluciones similares, análogamente el comportamiento que muestran al cambio en los parámetros es parecido. Para el caso de φ y q_0 se observa que el comportamiento de ACS converge en mayor tiempo cuando se aumenta la exploración con ciertos valores de los mismos. En conclusión, escoger el algoritmo a utilizar es tan importante como determinar los valores de los parámetros, ya que pequeños cambios en ellos pueden conducir a cambios significativos en los resultados finales.

Finalmente, vale la pena mencionar que a partir de este trabajo, los trabajos futuros que se desprenden de éste consiste en implementar los algoritmos de hormigas en todo el circuito CU y así, tener una propuesta completa de nuevas rutas que optimicen la distancia recorrida por los vehículos que se encargan de la recolección de residuos en dicha universidad. También es deseable que la aplicación de estas técnicas de optimización se extienda y se apliquen en la ciudad de México, así como se ha hecho en otras zonas urbanas del mundo.

Agradecimientos. Los autores agradecen el apoyo otorgado al proyecto PAPIIT con número IN107214, así como a la Dirección General de Obras y Conservación de la UNAM, a la Ing. Araceli Flores Soto y al Arq. José Guadalupe González Cruz por el apoyo brindado al aportar información sobre la planeación actual de la recolección de residuos en UNAM-CU. Finalmente, Beatriz A. Garro-Licón agradece al CONACYT por la beca posdoctoral otorgada.

Referencias

1. Bonomo, F., Durán, G., Larumbe, F., Marenco, J.: A method for optimizing waste collection using mathematical programming: a Buenos Aires case study. Waste Management & Research, vol. 30, no. 3, pp.311–324 (2012)
2. Dorigo, M., Birattari, M., Stützle, T.: Ant colony optimization-artificial ants as a computational intelligence technique. IEEE Comput. Intell. MAG, vol. 1, pp. 28–39 (2006)
3. Dorigo, M., Gambardella, L. M.: Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 53–66 (1997)
4. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: an autocatalytic optimizing process. Université Libre de Bruxelles (1991)

5. Dorigo M.: Optimization, Learning and natural algorithms. Ph.D Thesis, Dip. Electtronica e Informazion, Politecnico di Milano, Italy (1992)
6. Filipiak, K. A., Abdel-Malek, L., Hsieh, H.-N., Meegoda, J. N.: Optimization of municipal solid waste collection system: case study. Practice Periodical of Hazardous, Toxic, and Radioactive Waste Management, vol. 13, no. 3, pp. 210–216 (2009)
7. Hemmelmayr, V. et al.: A heuristic solution method for node routing based solid waste collection problems. Journal of Heuristics, vol. 19, no. 2, pp. 129–156 (2013)
8. Hornig, E.S., Fuentealba, N. R.: Modelo ACO para la recolección de residuos por contenedores. Revista chilena de ingeniería, vol. 17, no. 2, pp. 236–243 (2009)
9. Secretaría Administrativa, Dirección de Obras y Conservación, http://www.obras.unam.mx/Pagina/, Actualizado: 23 de mayo de 2015.
10. Ismail, Z., Loh, S.: Ant colony optimization for solving solid waste collection scheduling problems. Journal of mathematics and statistics, vol. 5, no. 3, pp.199–205 (2009)
11. Karadimas, N. V. et al.: Optimal solid waste collection routes identified by the ant colony system algorithm. Waste Management & Research, vol. 2, no. 2, pp. 139–147 (2007)
12. Kulcar, T.: Optimizing solid waste collection in Brussels. European Journal of Operational Research, vol. 90, no. 1, pp. 71–77 (1996)
13. Mansini, R. et al.: A linear programming model for the separate refuse collection service. Computers & Operations Research, vol. 25, no. 7, pp. 659–673 (1998)
14. Mourao, M., Almeida, M. T.: Lower-bounding and heuristic methods for a refuse collection vehicle routing problem. European Journal of Operational Research, vol. 121, no. 2, pp. 420–434 (2000)
15. Ogwueleka, T. C.: Route optimization for solid waste collection: Onitsha (Nigeria) case study. Journal of Applied Sciences and Environmental Management, vol. 13, no. 2, pp. 37–40 (2009)
16. Stützle, T., Hoos, H.H.: Improving the Ant System: A detailed report on the MAX–MIN Ant System. Technical Report AIDA–96–12, F.G. Intellektik, F.B. Informatik, T.U. Darmstadt, Germany (1996)