Optimización de rutas mediante computación bioinspirada, un paralelo entre hormigas artificiales y algoritmos genéticos

Route optimization through bioinspired computation, a parallel between artificial ants and genetic y algorithms

Sergio A. Rojas Galeano

Ingeniero de sistemas Universidad Nacional de Colombia. Especialista en Ingeniería de Software Universidad Distrital Francisco José de Caldas. Estudiante de MSc. Intelligent Systems, University of London. Docente de la Universidad Distrital Francisco José de Caldas adscrito a la Facultad de Ingeniería.

Clasificación del artículo: Reflexión.
Fecha de recepción: abril 16 de 2004.
Fecha de aceptación: junio 30 de 2004

Palabras clave: optimización con colonias de hormigas, algoritmos genéticos, problema agente viajero
Key words: ant colony optimization, genetic algorithms, traveling salesman problem

Resumen

Durante las últimas dos décadas los algoritmos genéticos han sido aplicados con éxito en la solución de problemas de optimización combinacional. Recientemente se ha propuesto como alternativa de solución un nuevo modelo de computación bioinspirada, conocido como optimización mediante colonias de hormigas. Los dos modelos comparten características tales como paralelismo, emergencia de complejidad e interacción entre múltiples agentes. En este artículo se describe un parangón entre las dos técnicas aplicadas a un problema de optimización típico: el problema del agente viajero. Los resultados revelan que para este caso las hormigas artificiales alcanzan soluciones superiores, lo que puede ser un indicador de la validez de su posible utilización en diversos campos.

Abstract

Genetic algorithms have been successfully applied for combinatorial optimization problems during the lasts two decades. A recently new approach for bioinspired computation known as Ant Colony Optimization has been reported. Both models are based on implicit parallelism and emergence of complex behaviors and interaction between multiple agents. In this paper we compare the two approaches related to a typical optimization problem: the problem of the traveling salesman. Results show that the artificial ants are able to obtain better outcomes. This may anticipate that they could be helpful to solve other kind of optimization problems.

1 Introducción

Los modelos de optimización combinacional (Castillo *et al*., 2002), mejor conocidos como programación matemática, se aplican a problemas de ingeniería que involucran usualmente una decisión sobre la mejor asignación de un conjunto de recursos a un conjunto de consumidores. Algunos ejemplos incluyen: los suministros que debe hacer un proveedor a sus distribuidores; la participación de las inversiones de una compañía en el mercado de valores; la asignación de los salones de una universidad al calendario de cursos semestral, o la distribución del presupuesto familiar para comprar el mercado semanal. La solución del pro-

blema consiste en una búsqueda sobre el espacio de todas las posibles combinaciones de las variables involucradas; cuando este número es relativamente pequeño, los problemas se tratan mediante las técnicas de programación lineal, entera y no lineal. Sin embargo, cuando el número de variables aumenta, también aumenta su complejidad computacional (en orden combinatorio o factorial), y es necesario aplicar técnicas aproximadas para encontrar una buena solución, aunque quizás no sea la óptima.

Recientemente se han propuesto como alternativa varios sistemas inteligentes cuya inspiración se fundamenta en mecanismos naturales. En particular, los algoritmos genéticos han tenido bastante aceptación y se han aplicado en varios casos con resultados exitosos (Gen y Cheng, 1999). Otra técnica novedosa, llamada optimización mediante colonias de hormiga, parece ser prometedora (Dorigo y Stützle, 2004). En este artículo se presenta un parangón entre los dos modelos empleados para solucionar un problema prototipo conocido como el "problema del agente viajero". Para probar el primer modelo se utilizó una librería de software preconstruida; para el segundo, se desarrollaron todos los algoritmos en Matlab, aprovechando las ventajas de programación vectorial ofrecidas por esta herramienta, de modo que el tiempo de ejecución de los experimentos fue optimizado. Dados los resultados, es posible concluir que las hormigas artificiales ganan el mérito para este problema, lo cual puede ser un buen indicador de su posible empleo eficaz en otros campos o aplicaciones.

El documento está organizado de la siguiente manera: en primer lugar se realiza un repaso teórico del problema y de las dos técnicas bioinspiradas; en seguida se explica el diseño experimental, luego se describen los resultados y, finalmente, se presentan las conclusiones y las expectativas de trabajo futuro.

2. Materiales y métodos

2.1 El problema del agente viajero

En términos coloquiales, el problema conocido como "del agente viajero" (Applegate *et al*., 1998) –en adelante TSP, por sus siglas en inglés– consiste en el aprieto en que se encuentra un representante de ventas cuyo jefe le ha ordenado distribuir las muestras de su producto en n ciudades de forma que minimice el costo de los tiquetes de transporte entre ellas, empleando además el menor tiempo posible. En el presente estudio se optimizarán el costo y el tiempo con respecto a la distancia entre las ciudades, sabiendo que es posible viajar desde cualquier ciudad a todas las demás, y que la distancia de ir del origen i al destino j puede ser distinta a la distancia para ir de *j* a *i*.

La descripción del TSP puede realizarse utilizando un grafo como el mostrado en la figura 1; los vértices representan las ciudades, y los arcos tienen asociados las distancias entre ellas. Si se designa la matriz $[d_{ij}]$ para almacenar las distancias, y la matriz binaria $[x_{ij}]$ para representar una instancia de la solución del problema (en donde $x_{ij} = 1$ significa que la ruta incluye visitar la ciudad j despues de la i), el planteamiento del problema se resume en (1):

$$\begin{aligned} mín \quad & \sum_{i,j} d_{ij} x_{ij}, \\ s.a. \quad & \sum_i x_{ij} = 1, \\ & \sum_j x_{ij} = 1, \\ & x_{ij} \in \{0,1\} \end{aligned} \tag{1}$$

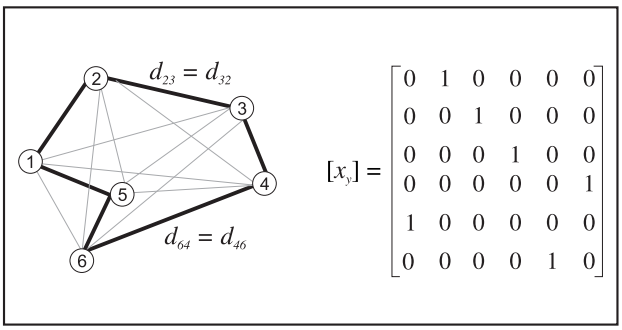

Figura 1. Grafo que representa un TSP de seis ciudades (izquierda). En línea resaltada se muestra una posible instancia de solución al problema; la matriz correspondiente (derecha) representa la ruta: 1~2~3~4~6~5~1

Los datos utilizados en los experimentos descritos en este artículo fueron tres instancias bien conocidas para el TSP (TSPLIB, 1995), correspondientes a 17, 48 y 100 ciudades[1]. El problema de 17 ciudades es simétrico, es decir $[d_{ij}] = [d_{ji}]$, mientras que los otros dos son asimétricos.

[1] Los tres conjuntos de datos pueden obtenerse respectivamente de:

17: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/atsp/br17.atsp.gz

48: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/atsp/ry48p.atsp.gz

100: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/atsp/kro124p.atsp.gz

2.2 Algoritmos genéticos

Los algoritmos genéticos (Mitchell, 1998) –en adelante GA, por sus siglas en inglés– mantienen una población de individuos representados por un genoma (una cadena de caracteres binarios o decimales) que codifica una posible solución al problema. Esta población es sometida a un proceso de evolución artificial guiado por una presión selectiva hacia los candidatos más fuertes, más aptos, o, en otras palabras, los que representen soluciones óptimas. Durante este proceso los individuos, creados inicialmente al azar, son sometidos a diferentes operaciones que introducen variaciones en su contenido genético (recombinación o cruce, mutaciones), dando lugar a nuevas soluciones que, en promedio, y a medida que avanzan las generaciones, poseen una mayor aptitud. Tanto el tamaño de la población como la longitud del genoma permanecen constantes. La solución final del problema estará dada por el individuo más apto encontrado en todas las generaciones; por ejemplo, para el problema TSP de seis ciudades de la figura 1, una población de cuatro individuos con un genoma codificado en números enteros podría ser:

[1 2 3 4 5 6]

[3 1 5 6 4 2]

[1 6 3 2 5 4]

[5 2 3 6 4 1]

El *k*-ésimo genoma tiene una correspondencia uno a uno con una matriz x_{ij}^{k}, como la mostrada en la misma figura. La aptitud del *k*-ésimo individuo es inversamente proporcional a la distancia total recorrida por esa ruta (2):

$$f_k = \left(\sum_{i,j} d_{ij} x_{ij}^{k} \right)^{-1} \qquad (2)$$

Existen numerosos tipos de operaciones genéticas en los GA. Las más comúnmente utilizadas son:

- *Selección proporcional a la aptitud (método de la ruleta).* Cada individuo es evaluado de acuerdo con la función de aptitud mostrada en (2). En proporción a su aptitud, a cada individuo se asigna un porcentaje del total de números de una ruleta simulada. La ruleta se rueda tantas veces como individuos hay en la población; la siguiente generación la conforman los individuos que resultan ganadores. De esta forma, aquellos con mayor aptitud (los más fuertes) tienen ventaja sobre los de menor, y probablemente sobrevivirán, mientras sus contrincantes probablemente se extinguirán. La porción de ruleta asignada al k-ésimo individuo (y en consecuencia, su probabilidad de ser seleccionado) está dada por (3), donde N es el tamaño de la población:

$$s_k = \frac{f_k}{\sum_{n=1}^{N} f_n} \qquad (3)$$

- *Recombinación o cruce.* Se seleccionan dos padres al azar y, de acuerdo con una probabilidad de cruce, su contenido genético es recombinado para producir dos descendientes. La teoría de la evolución plantea que, en la medida en que los padres son aptos, también lo serán los hijos, los cuales en promedio llegan a superar a sus progenitores. La selección y la mutación son los operadores que permiten amplificar las buenas soluciones parcialmente encontradas. Para el TSP es posible plantear muchos tipos de recombinación; por ejemplo, tomando los dos últimos individuos de la población mostrada anteriormente, puede generarse la siguiente descendencia (los puntos de cruce están marcados con 'l'):

[1 6 |**3 2 5**| 4] [|***3 6 4***|] [1 2 |***3 6 4***| 5]

X => =>

[*5 2* |***3 6 4***| *1*] [|**3 2 5**|] [*6 4* |**3 2 5**| *1*]

Nótese que el primer paso consiste en intercambiar las ciudades entre los puntos de cruce; luego se procede a rellenar los espacios vacíos con los genes de los padres que no estén repetidos en la subcadena recién cruzada, manteniendo el orden original.

- *Mutación.* Esporádicamente se elige un individuo de la población y se modifica uno de los genes por un valor aleatorio. Este operador permite al GA explorar candidatos no evaluados dentro del espacio de soluciones del problema, que posiblemente representen mejores soluciones que las evolucionadas hasta una generación determinada. En el caso del TSP, una mutación consiste en elegir dos ciudades al azar y permutarlas en las respectivas posiciones del genoma; por ejemplo, el segundo

hijo con una mutación en la tercera y sexta posición daría lugar a:

```
[6 4 3 2 5 1] => [6 4 1 2 5 3]
```

2.3 Optimización mediante colonias de hormigas

La colonias de hormigas demuestran comportamientos que pueden ser calificados como asombrosos e ingeniosos a la vez: la maravillosa arquitectura de un hormiguero o la forma como arman filas durante la búsqueda de comida son algunos ejemplos. En particular, la causa de la formación de filas en el camino que une el hormiguero con la fuente de comida, puede resumirse así: cuando una hormiga sale de su nido va dejando un rastro de sustancias químicas llamadas feromonas, las cuales le sirven como marcas en el trayecto. Eventualmente, otras hormigas pueden descubrir el camino marcado por algunas de sus compañeras, ya que son más propensas a seguir por rutas anteriormente impregnadas de feromonas, que a vagar por trayectorias inexploradas. Luego de un período de tiempo en el que las hormigas repiten este comportamiento, emerge una ruta claramente definida del hormiguero hacia la comida, seguido por todas las hormigas. Lo más interesante es que se ha comprobado experimentalmente que dicha ruta resulta ser la más corta, es decir, la óptima (Bonabeau *et al.*, 2000). Además, ante la aparición imprevista de un obstáculo en una ruta óptima anteriormente descubierta, las hormigas se las arreglan para encontrar una nueva mejor ruta (figura 2).

En Dorigo *et al* (1997 y 1999) se propone un modelo basado en este comportamiento para resolver el TSP, llamado *optimización mediante colonias de hormigas* –ACO, por sus siglas en inglés–. La idea consiste en utilizar una población de hormigas artificiales ubicadas en forma aleatoria en los diferentes vértices del grafo del TSP. Cada hormiga debe completar un circuito moviéndose paso a paso a través de los arcos para, al terminar, depositar una cantidad de feromonas inversamente proporcional a la distancia recorrida. De esta forma se favorecen las rutas más cortas. La selección de la próxima ciudad a visitar es estocástica, sesgada por la concentración de feromonas que encuentra en cada arco. A diferencia de las hormigas naturales, las artificiales tienen una memoria en donde almacenan el circuito realizado. La memoria de cada hormiga representa una posible solución, y la solución definitiva estará dada por la hormiga que haya descubierto el camino más corto.

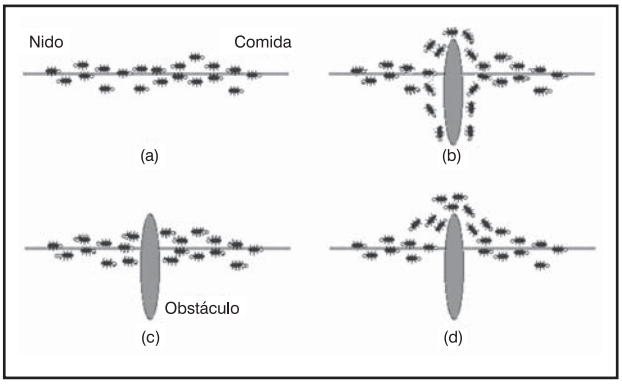

Figura 2. Re-descubrimiento del camino más corto realizado por las hormigas[2]. (a) La ruta más corta entre el nido y la comida ha emergido como resultado del proceso de comunicación con feromonas entre las hormigas; (b) un obstáculo se interpone en el camino; (c) un grupo de hormigas elige, en forma aleatoria, el camino más corto (otras el más largo), dejando a su paso nuevas pistas de feromonas; (d) puesto que por el camino más corto las hormigas viajan más rápido, al cabo de un tiempo éste se impregnará con una mayor cantidad de feromonas, y por tanto las hormigas preferirán tomarlo, evitando así la ruta más larga

En este modelo, la búsqueda de nuevas alternativas y el mejoramiento de las ya encontradas se realizan usando los operadores para seleccionar la siguiente ciudad por visitar, y para diseminar las hormonas:

- Movimiento de la colonia (o selección de la siguiente ciudad por visitar). Las hormigas pueden ser de dos tipos: exploradoras o explotadoras; las exploradoras se aventuran por caminos poco transitados, mientras que las explotadoras prefieren seguir los trayectos con mayor concentración de feromonas. Si la hormiga es explotadora, entonces selecciona la siguiente ciudad por visitar de acuerdo con la ecuación (4), en donde u toma como valores las ciudades que aún no han sido visitadas por la hormiga (o que no se encuentran en su memoria M_k), $\tau\ (r,\ u)$ es la concentración de feromonas presente en el camino que une a la ciudad r con u, y $\eta(r,\ u)$ es el inverso de la distancia entre r y u:

$$s = \arg\max_{u \notin M_k} [\tau(r,u) * \eta(r,u)] \qquad (4)$$

De otra forma, la hormiga selecciona explorar una ciudad no visitada de acuerdo con la probabilidad dada por (5):

$$p_s = \frac{[\tau(r,u) * \eta(r,u)]}{\sum_{u \notin M_k} [\tau(r,u) * \eta(r,u)]} \qquad (5)$$

[2] Esta figura ha sido adaptada de Dorigo y Gambardella, 1997.

Así, los caminos que tengan mayor cantidad de feromonas entre distancias cortas serán preferidos por las hormigas. Nótese que el resultado de $\tau(r, u)*\eta(r, u)$ puede ser considerado como la aptitud de una determinada ciudad, con lo cual la ecuación (5) mantiene una gran similitud con el método de selección por la ruleta mostrado en (3).

- *Diseminación de las feromonas.* La concentración de feromonas en los arcos que comunican dos ciudades puede cambiar debido a un factor local que simula la evaporación que ocurre naturalmente, o debido a un factor global que simula la segregación de hormonas que realiza la hormiga triunfadora, es decir, la que solamente impregna los arcos correspondientes al trayecto más corto en una instancia determinada. La evaporación se simula mediante la ecuación (6), en donde τ_0 es un parámetro correspondiente a la longitud del recorrido dada por la heurística del vecino más cercano, y $\alpha = 0{,}1$ es el factor de evaporación.

$$\tau(r,s) = (1-\alpha)\tau(r,s) + \alpha\tau_0 \qquad (6)$$

La segregación de feromonas la realiza únicamente la hormiga ganadora mediante la ecuación (7), en donde $\Delta\eta$ es el inverso del circuito más corto, es decir, de la distancia recorrida por dicha hormiga.

$$\tau(r,s) = (1-\alpha)\tau(r,s) + \alpha\Delta\eta \qquad (7)$$

2.4 Diseño experimental

Para ilustrar el desempeño del GA en la solución de los problemas anteriormente mencionados, se utilizó la librería GAOT para Matlab[3]. El cromosoma se codificó con una cadena de números enteros donde cada gen representa el número de la ciudad visitada en el orden respectivo. Se utilizaron diferentes tipos de cruce (OX, PMX), mutaciones (SWAP, ADJSWAP, SHIFT) y método de selección por ruleta normalizada (normSelect), los cuales son variaciones de los operadores descritos anteriormente (explicados en detalle por Houck *et al*., 1995). Respecto a los parámetros de ejecución del GA (tamaño de la población, tasa de cruce, mutación, selección y número de generaciones) se realizaron múltiples experimentos para escoger los valores más apropiados. En la sección de resultados se muestran dos experimentos representativos para cada problema, con los correspondientes parámetros.

[3] La librería GAOT es de libre utilización y está disponible en http://www.ie.ncsu.edu/mirage/GAToolBox/gaot/

Seguidamente, para completar el cuadro comparativo con el desempeño del ACO en la solución de los mismos problemas, se diseñó e implementó una librería para la ejecución de ACO en Matlab. El algoritmo básico desarrollado se muestra en la figura 3. Todas las operaciones allí mostradas fueron implementadas de acuerdo con la descripción dada anteriormente; sin embargo, teniendo en cuenta las características de paralelismo implícito con que el que operan los múltiples agentes (colonia de hormigas), se aprovechó el procesamiento vectorial ofrecido por la máquina virtual de Matlab para realizar una elaboración mucho más eficiente, reduciendo así el tiempo de ejecución de los experimentos. De manera similar a como se procedió con el GA, los parámetros de ejecución del ACO (proporción de hormigas explotadoras, tamaño de la colonia, número máximo de recorridos) se escogieron como los más eficaces, luego de realizar varios experimentos. En la siguiente sección se detallan, junto con los resultados obtenidos.

```
Algoritmo_ACO_TSP
  REPEAT UNTIL num_max_recorridos
      WHILE NOT(todas_ciudades_visitadas)
        Mover_un_paso(Colonia)
        Actualizacion_local_feromonas
    END
    Calcular_longitud_caminos_recorridos (Colonia)
  Actualizacion_global_feromonas(mejor_hormiga(Colonia))
END
RETURN mejor_hormiga_jamas(Colonia)
```

Figura 3. Pseudocódigo del algoritmo implementado para la solución del TSP usando ACO

3. Resultados y discusión

La mejor solución conocida hasta el momento para el TSP de 17 ciudades tiene una longitud de 39 unidades[4]. La tabla 1 muestra los resultados obtenidos utilizando GA y ACO; ambos enfoques obtienen una solución con esta longitud, lo cual era de esperarse considerando que se trata de un problema con un número pequeño de ciudades. Sin embargo, es de resaltar que no todas las veces que se ejecutó el GA se obtuvo la solución óptima (como se ve en el primer resultado, *longitud* = 40); por su parte, el ACO se mostró más robusto ya que siempre alcanzó una *longitud* = 39, incluso ensayando parámetros bastante

[4] *Véase* www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/doc.ps

disímiles, como la tasa de exploración, que para la primera ejecución mostrada en la tabla fue de 0,8, y de 0,2 para la segunda. Es posible encontrar múltiples soluciones óptimas al TSP17 (circuitos con ciudades en diferente orden), dado que la matriz de distancias entre ciudades es simétrica.

Para el caso del TSP de 48 ciudades, 14.422 es la menor distancia conocida hasta ahora[5]. La tabla 2 contiene los resultados de los experimentos realizados. La mejor solución alcanzada con el GA fue 19.278 utilizando 50 individuos durante 500 generaciones; esto significa un desfase con el óptimo de un 33%. Por su parte el ACO, con 50 hormigas durante sólo 100 recorridos (generaciones), alcanzó una distancia de 14.814: apenas un 2% por encima del mínimo. Es decir, la efectividad alcanzada por las hormigas fue muy superior para un problema de mediana envergadura.

Tabla 1. Parámetros y resultados del GA y ACO para el TSP de 17 ciudades

GA	Operadores genéticos:	cruce OX, mutación SWAP
	Tamaño de la población:	300
	Tasa de cruce:	86%
	Tasa de mutación:	5%
	Tasa de selección:	30%
	Núm. de generaciones:	10
	Solución obtenida:	14∼2∼11∼9∼17∼8∼4∼5∼15∼7∼6∼16∼1∼12∼10∼13∼3
	Longitud:	40
	Operadores genéticos:	cruce OX, mutación SWAP
	Tamaño de la población:	1000
	Tasa de cruce:	80%
	Tasa de mutación:	20%
	Tasa de selección:	30%
	Núm. de generaciones:	10
	Solución obtenida:	3∼1∼12∼8∼9∼17∼5∼4∼16∼6∼15∼7∼2∼13∼11∼10∼14
	Longitud:	39
ACO	Tamaño de la población:	20
	Tasa de exploración:	0,8
	Máx.núm. de recorridos:	10
	Solución obtenida:	14∼3∼2∼10∼13∼11∼7∼6∼16∼15∼4∼5∼17∼8∼9∼1∼12
	Longitud:	39
	Tamaño de la población:	20
	Tasa de exploración:	0,2
	Máx.núm. de recorridos:	10
	Solución obtenida:	2∼11∼10∼13∼14∼3∼12∼1∼8∼9∼17∼4∼5∼7∼6∼16∼15
	Longitud:	39

Tabla 2. Parámetros y resultados del GA y ACO para el TSP de 48 ciudades

GA	Operadores genéticos:	cruce OX, PMX, mutación SWAP, ADJSWAP, SHIFT
	Tamaño de la población:	100
	Tasa de cruce:	70%
	Tasa de mutación:	30%
	Tasa de selección:	30%
	Núm. de generaciones:	500
	Solución obtenida:	33∼20∼13∼25∼48∼5∼42∼29∼41∼16∼1∼44∼18∼7∼36∼30∼28∼31∼8∼22∼3∼34∼2∼26∼4∼39∼14∼23∼12∼15∼27∼17∼43∼37∼19∼6∼47∼21∼32∼24∼45∼35∼10∼11∼40∼9∼38∼46
	Longitud:	20.600
	Operadores genéticos:	cruce OX, PMX, mutación SWAP, ADJSWAP, SHIFT
	Tamaño de la población:	50
	Tasa de cruce:	48%
	Tasa de mutación:	30%
	Tasa de selección:	30%
	Núm. de generaciones:	500
	Solución obtenida:	39∼48∼42∼26∼4∼2∼29∼14∼13∼23∼15∼6∼30∼43∼28∼36∼46∼11∼47∼20∼18∼7∼37∼19∼27∼17∼44∼31∼33∼12∼32∼24∼10∼45∼35∼5∼25∼40∼9∼38∼1∼8∼22∼16∼3∼41∼34∼21
	Longitud:	19.278
ACO	Tamaño de la población:	60
	Tasa de exploración:	0,9
	Máx.núm. de recorridos:	50
	Solución obtenida:	14∼34∼3∼22∼16∼41∼29∼42∼10∼24∼45∼35∼26∼4∼2∼5∼48∼39∼32∼21∼47∼20∼33∼46∼15∼40∼9∼1∼8∼38∼31∼44∼36∼18∼7∼28∼37∼19∼27∼17∼6∼30∼43∼12∼11∼23∼13∼25∼14
	Longitud:	15.065
	Tamaño de la población:	50
	Tasa de exploración:	0,9
	Máx.núm. de recorridos:	100
	Solución obtenida:	41∼34∼29∼2∼26∼4∼35∼45∼10∼24∼42∼5∼48∼39∼32∼21∼12∼15∼40∼9∼1∼8∼38∼31∼44∼36∼30∼43∼17∼19∼27∼6∼37∼28∼7∼18∼46∼33∼20∼47∼13∼25∼14∼23∼11∼3∼22∼16∼41
	Longitud:	14.814

Finalmente, los resultados para el TSP de 100 ciudades se resumen en la tabla 3.

[5] Ibid.

Tabla 3. Parámetros y resultados del GA y ACO para el TSP de 100 ciudades

GA	Operadores genéticos:	cruce OX, PMX, mutación SWAP, ADJSWAP, SHIFT
	Tamaño de la población:	100
	Tasa de cruce:	50%
	Tasa de mutación:	15%
	Tasa de selección:	30%
	Núm. de generaciones:	1000
	Solución obtenida:	62∼35∼57∼58∼54∼40∼89∼42∼92∼8∼23∼91∼99∼65∼66∼26∼19∼75∼56∼31∼97∼84∼36∼59∼74∼28∼64∼2∼50∼44∼68∼85∼30∼52∼78∼71∼100∼41∼48∼46∼34∼12∼27∼77∼1∼72∼21∼81∼61∼69∼82∼95∼33∼73∼25∼98∼17∼15∼47∼6∼49∼90∼53∼79∼94∼22∼16∼10∼63∼11∼32∼45∼55∼43∼3∼29∼14∼96∼76∼13∼5∼37∼39∼93∼38∼24∼18∼88∼70∼4∼80∼67∼9∼83∼7∼51∼87∼86∼20∼60
	Longitud:	63.145
	Operadores genéticos:	cruce OX, PMX, mutación SWAP, ADJSWAP, SHIFT
	Tamaño de la población:	200
	Tasa de cruce:	80%
	Tasa de mutación:	37%
	Tasa de selección:	30%
	Núm. de generaciones:	500
	Solución obtenida:	40∼2∼95∼82∼39∼29∼71∼100∼41∼48∼78∼52∼76∼13∼33∼37∼5∼96∼14∼46∼34∼9∼60∼98∼84∼10∼72∼74∼59∼21∼17∼12∼55∼43∼3∼30∼44∼64∼67∼42∼56∼63∼11∼15∼32∼91∼54∼50∼73∼25∼87∼61∼58∼8∼75∼19∼94∼88∼53∼90∼47∼20∼27∼35∼62∼86∼83∼85∼68∼69∼81∼93∼28∼92∼65∼26∼4∼66∼99∼24∼38∼36∼45∼23∼77∼57∼7∼51∼1∼79∼18∼70∼22∼16∼49∼6∼97∼89∼31∼80
	Longitud:	57.463
ACO	Tamaño de la población:	70
	Tasa de exploración:	0,9
	Máx.núm. de recorridos:	80
	Solución obtenida:	34∼29∼3∼43∼46∼14∼71∼100∼41∼48∼30∼52∼5∼37∼96∼78∼39∼82∼13∼76∼33∼95∼2∼64∼40∼54∼44∼50∼73∼69∼25∼81∼61∼51∼87∼9∼7∼12∼55∼83∼57∼62∼35∼20∼27∼86∼60∼77∼98∼11∼17∼15∼32∼45∼23∼91∼21∼59∼36∼99∼24∼38∼84∼10∼72∼74∼47∼63∼6∼49∼90∼53∼79∼18∼94∼22∼16∼88∼70∼66∼65∼26∼4∼19∼75∼97∼89∼42∼56∼80∼31∼8∼92∼1∼93∼28∼67∼58∼68∼85∼34
	Longitud:	40.368
	Tamaño de la población:	200
	Tasa de exploración:	0,8
	Máx.núm. de recorridos:	300
	Solución obtenida:	34∼29∼30∼52∼5∼37∼33∼76∼13∼95∼82∼50∼73∼69∼54∼64∼40∼2∼44∼68∼85∼39∼96∼78∼48∼100∼41∼71∼14∼46∼3∼43∼55∼83∼57∼7∼12∼27∼86∼20∼77∼35∼62∼60∼47∼32∼45∼23∼91∼98∼11∼17∼15∼74∼59∼21∼72∼10∼84∼99∼38∼36∼24∼18∼79∼90∼92∼8∼42∼89∼31∼80∼97∼4∼66∼65∼26∼70∼94∼88∼22∼16∼53∼19∼75∼56∼63∼49∼6∼1∼93∼28∼67∼58∼25∼81∼61∼51∼87∼9
	Longitud:	38.319

Existe un óptimo conocido de 36.230 para este problema[6]; ninguno de los dos métodos experimentados alcanzó un nivel equivalente. Sin embargo, cabe resaltar que mientras la mejor solución obtenida por el GA estuvo por debajo del óptimo en casi un 60% (57.463), al ACO estuvo solo a un 5% (38.319) de alcanzarlo. Además, debido a limitaciones de cómputo, la mayoría de veces el ACO fue ejecutado con un número menor de individuos y de generaciones que el GA, lo cual es un vestigio de la posibilidad de encontrar aún mejores soluciones en caso de contar con recursos más potentes.

4. Conclusiones y trabajo futuro

El enfoque de computación bioinspirada, conocido como ACO, representa una nueva alternativa válida para atacar problemas de optimización que, por su naturaleza combinatoria, tienden a ser difíciles de resolver para instancias de mediana y gran envergadura. En este artículo se presentó un estudio comparativo entre ACO y GA; los resultados recopilados muestran que, por lo menos en esta contienda, el hormiguero artificial superó con creces a la evolución simulada (a pesar de que los dos enfoques presentan similitudes).

El ACO para un TSP de 100 ciudades alcanzó una solución tan sólo 5% por debajo del óptimo conocido, mientras que el GA fue inferior en un 60%, por lo que puede considerarse al primero como un mecanismo viable para la búsqueda de soluciones aceptables en problemas comparables. De hecho, Rossi-Doria *et al.* (2003) han mostrado estudios similares donde comparan la efectividad de ACO, GA y otras técnicas de búsqueda en instancias diferentes de TSP y en otros problemas de programación lineal y entera, encontrando igualmente resultados favorables para las hormigas. Un aspecto de atención para futura investigación puede ser la utilización de la dos técnicas combinadas, en donde soluciones encontradas simultáneamente por el AG y por ACO pueden mezclarse con el fin de favorecer la optimización (Acan, 2002). También puede ser interesante investigar la utilización de una técnica para afinar la otra, por ejemplo, usar un AG para optimizar los parámetros de ejecución de un ACO.

Aunque se trata de una propuesta reciente, el ACO es uno de los temas de mayor inquietud académica dentro de la comunidad dedicada al estudio de los sistemas inteligentes, razón por la cual existen grandes e interesantes retos para continuar con trabajos futuros en esta área. Puede pensarse, por ejemplo, en extrapolar el planteamiento formulado para el TSP a otros problemas combinatorios, como la asignación, programación y distribución de recursos mencionados en la sección introductoria; por otra parte, podría utilizarse

[6] Ibid.

en aplicaciones reales como problemas de enrutamiento de paquetes en redes de comunicaciones (Bonabeau *et al*., 2000) o distribución de productos en estanterías de supermercados. En esencia, el modelo representado por ACO podría ser aplicado a cualquier problema en donde se requiera encontrar una solución óptima dentro de un espacio de búsqueda bastante grande. Por este motivo, el potencial de aplicación puede ser enorme[7] y, al parecer, las hormigas artificiales prometen ser tan simpáticas e ingeniosas como su contraparte natural.

Referencias bibliográficas

[1] Acan, A. (2002). GAACO: A GA + ACO Hybrid for Faster and Better Search Capability. Ant Algorithms, Third International Workshop, ANTS 2002, Brussels, Belgium, September 12-14; 300-301.

[2] Applegate, D., Bixby, R, Chvatal, V, Cook, W. (1998). On the Solution of Traveling Salesman Problems. *Documenta Mathematica*, Extra Volume ICM, III, 645-656.

[3] Bonabeau, E., Dorigo, M. and Theraulaz, G. (2000). "Inspiration for Optimization from Social Insect Behaviour". In: *Nature*. Vol. 406: 39-42.

[4] Castillo, E., Conejo, A. J., Pedregal, P., García, R., Alguacil, N. (2002). *Building and Solving Mathematical Programming Models in Engineering and Science*. New York, Wiley.

[5] Dorigo M. y Gambardella, L. (1997). "Ant Colonies for the Traveling Salesman Problem". In: *BioSystems*, 43;73-81.

[6] Dorigo M, Di Caro, G, Gambardella, L. (1999). "Ant Algorithms for Discrete Optimization". In: *Artificial Life*, 5(2);137-172.

[7] Dorigo, M. y Stützle, T. (2004). Ant Colony Optimization. Bradford Books.

[8] Gen, M. y Cheng, R. (1999). *Genetic Algorithms and Engineering Optimization*. New York,Wiley.

[9] Houck, C., Joines, J., Kay, M. (1995). A Genetic Algorithm for Function Optimization: A Matlab Implementation. North Carolina State University, NCSU-IE TR 95-09.

[10] Mitchell, M. (1998 Reprint Ed.). *An Introduction to Genetic Algorithms*. MIT Press.

[11] Rossi-Doria O., Sampels, M., Birattari, M., Chiarandini, M., Dorigo, M., Gambardella, L., Knowles, J., Manfrin, M., Mastrolilli, M., Paechter, B., Paquete, L., Stützle, T. (2003). "A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem".In: *Practice and Theory of Automated Timetabling IV, Lecture Notes in Computer Science* 2740; 329-351, Berlin Heidelberg, Springer-Verlag.

[12] TSPLIB (1995). A library of sample instances for the TSP. Disponible en world wide web: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

[7] Para profundizar en los temas de investigación de punta en ACO, se recomienda visitar el sitio web de la conferencia mundial que se realiza cada bienio: http://iridia.ulb.ac.be/%7Eants/ants2004/index.html