Evaluación de algoritmos híbridos basados en colonias de hormigas para TSP

Carlos Grandón, Romina Torres, María Cristina Riff
e-mail: {cobra, romina, mcriff}@inf.utfsm.cl

Abstract

Los problemas de optimización combinatoria han mostrado características de "intratabilidad" al intentar ser resueltos por métodos tradicionales. Debido a que este tipo de problemas se presentan en un gran número de aplicaciones del mundo real, es que investigadores han propuesto métodos alternativos que tratan de encontrar una solución aproximada al problema. Dentro de este tipo de métodos se encuentran los de búsqueda local.

En la búsqueda de nuevas estrategias, se han propuesto algoritmos que utilizan como base la metáfora de los insectos sociales. En particular, los algoritmos basados en colonias de hormigas han sido aplicados con éxito en algunas instancias del problema del vendedor viajero. En el presente artículo se presentan algoritmos híbridos basados en colonias de hormigas. El primer grupo de algoritmos trabaja sobre Ant System (AS). Estos algoritmos incluyen técnicas de búsqueda local 2-opt, 3-opt hill-climbing. El segundo grupo incorpora a ACS una nueva noción de contaminación para manejar el estancamiento en un óptimo local, e incluye además una estrategia GRASP. Los resultados mostraron un mejor desempeño con respecto a los algoritmos existentes en la literatura, en el marco del problema del vendedor viajero.

1 Introducción

Los problemas de optimización combinatoria han mostrado características de "intratabilidad" al intentar ser resueltos por métodos tradicionales. Debido a que este tipo de problemas se presentan en un gran número de aplicaciones del mundo real, es que investigadores han propuesto métodos alternativos que tratan de encontrar una solución aproximada al problema. En la búsqueda de nuevas estrategias, nacen algoritmos que utilizan como base la metáfora de los insectos sociales. En particular, los algoritmos basados en colonias de hormigas han sido aplicados con éxito en algunas instancias del problema del vendedor viajero. Se sostiene que las hormigas poseen un sistema de organización empleado para la búsqueda de comida. Este sistema utiliza una hormona llamada "feromona", que sirve como medio de comunicación entre individuos de la colonia. Cuando una hormiga se enfrenta a la elección de una ruta a seguir, ella es influenciada principalmente por la cantidad de feromona existente en aquella ruta. Si la cantidad en distintas rutas fuese la misma (o no hubiere feromona) el movimiento realizado por la hormiga sería aleatorio. Por su parte, desde el momento en que la hormiga escoge una ruta a seguir, ésta deposita feromona sobre el camino recorrido para orientar a las demás a seguirla.
La idea del por qué este sistema funciona se muestra en la figura 1. Un grupo de hormigas posee una ruta hacia el alimento (A). En un momento dado, se interpone un obstáculo en el camino. Existen básicamente dos alternativas a seguir: hacia la izquierda o hacia la derecha. En principio, ambas poseen la misma probabilidad de ser elegidas (dado que no existen cantidades de feromona hacia ambos lados), por tanto es razonable pensar que un grupo equivalente de hormigas utilizará ambas rutas (B). Finalmente, las hormigas que toman el camino más corto llegan primero a la ruta original, produciendo una acumulación mayor de feromona sobre esa ruta (C). Por esta razón, las próximas hormigas que lleguen al obstáculo estarán más propensas a escoger la ruta más corta (mayor feromona) que la ruta alternativa, puesto que en esta última el nivel de feromona será menor.

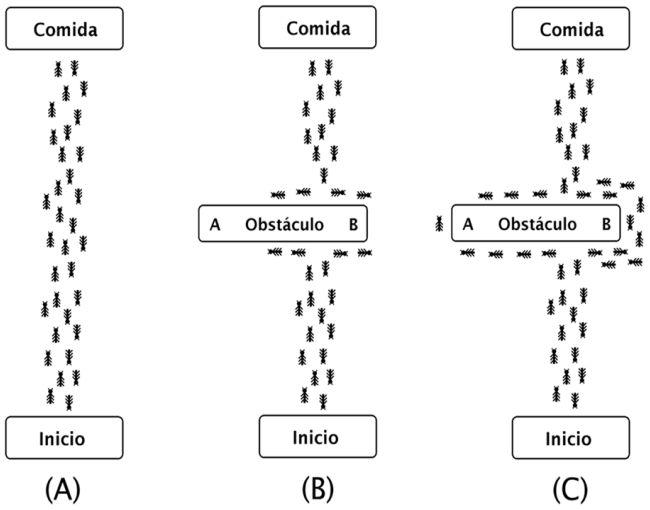

Figure 1: *Explicación del funcionamiento del sistema en hormigas.*

En la siguiente sección se presentarán los conceptos que rigen un algoritmo basado en hormigas, en la sección 3 se presenta el algoritmo Ant System (AS) propuesto por Marco

Dorigo. En la sección 4 se presentan algunas mejoras introducidas a AS al incorporarle técnicas de búsqueda local y se muestran los resultados para un conjunto seleccionado de problemas conocidos en la literatura. En la sección 5 se describe el algoritmo Ant Colony System (ACS) propuesto por Marco Dorigo y Luca Gambardella. En la sección 6 se presentan mejoras introducidas a ACS al incorporar nociones de contaminación, así como la inclusión de un algoritmo GRASP para la selección de ciudades. Además, se muestran en esta sección los resultados obtenidos para un conjunto de problemas reconocidos en la literatura. Finalmente, en la sección 7 se presentan las conclusiones y proyecciones futuras de esta investigación.

2 Ideas Básicas en los algoritmos basados en hormigas:

- Uso de un mecanismo de feedback positivo: reforzamiento de buenos caminos o soluciones
- Una feromona virtual que permite mantener en memoria las mejores opciones
- El riesgo de óptimo local o *stagnation* se controla por un mecanismo de feedback negativo denominado evaporación de la feromona que se maneja en unidades de tiempo. Este tiempo no debe ser muy grande porque produciría la caída en un óptimo local, ni tan reducida que anule el comportamiento cooperativo del algoritmo.
- Comportamiento cooperativo: las hormigas realizan exploración simultánea de soluciones diferentes realizado por un conjunto de hormigas iguales. Las hormigas que realizan bien su trabajo influyen el comportamiento de futuras hormigas. Ya que las hormigas exploran varias quasi-soluciones, la feromona resultante es la consecuencia de diferentes perspectivas del espacio de soluciones o de búsqueda. Aún cuando sólo se permite a la mejor hormiga reforzar la solución, existe un efecto cooperativo en el tiempo, porque las hormigas de la siguiente iteración usan la feromona para guiar su exploración.

3 Ant System (AS)

En el vendedor viajero el objetivo es encontrar el tour más corto para cubrir n ciudades. Cada ciudad debe ser visitada una sola vez. En el espacio euclideano la distancia d_{ij} entre la ciudad i y la ciudad j se expresa como:

$$d_{ij} = [(x_i - x_j)^2 + (y_i - y_j)^2]^{\frac{1}{2}} \tag{1}$$

El grafo $G = (N, E)$ no requiere estar completamente conectado, ni tampoco ser simétrico.

Sea $b_i(t), (i = 1, .., n)$ el número de hormigas en la ciudad i en el instante t. Sea $m = \sum_{i=1}^{n}$ el número total de hormigas. Sea $\tau_{ij}(t + z)$ la intensidad de la feromona de la conección (ij) en el instante $t + z$ con:

$$\tau_{ij}(t + z) = (1 - \rho)\tau_{ij}(t) + \Delta\tau_{ij}(t + z) \tag{2}$$

donde $0 < \rho \leq 1$ es un coeficiente que representa la evaporación. La cantidad de feromona dejada en el arco (ij) por la k-ésima hormiga en el instante $t + z$ está dada por:

$$\Delta\tau_{ij}^{k}(t + z) = \begin{cases} \frac{Q}{L_k} & \text{si la k-ésima hormiga usa el arco } (ij) \\ 0 & \text{e.o.c} \end{cases} \tag{3}$$

donde Q es una constante y L_k es el largo del tour encontrado por la k-ésima hormiga. La intensidad inicial de cada arco $\tau_{ij}(0)$ es casi cero.

La regla de transición, o la probabilidad que la hormiga k en la ciudad i vaya a la ciudad j es:

$$P_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{h \in permitido_k(t)} [\tau_{ih}(t)]^\alpha [\eta_{ih}]^\beta} & \text{si } j \in permitido_k(t) \\ 0 & \text{e.o.c.} \end{cases} \quad (4)$$

donde $permitido_k(t)$ es el conjunto de ciudades no visitadas por la hormiga k hasta el instante t y η_{ij} representa una heuristica local. Para el vendedor viajero, la heuristica más comúnmente utilizada es $\eta_{ij} = \frac{1}{d_{ij}}$, denominada la *visibilidad*. Los parámetros α y β entregan la importancia relativa entre la feromona y la visibilidad. Además, cada hormiga tiene asociada una lista tabú, la cual incluye todas las ciudades ya visitadas por la hormiga. El algoritmo AS se muestra en la figura 2.

Procedure Ant System
Inicializar
Para t=1 **hasta** el número de ciclos **haga**
Inicio
Para k = 1 **hasta** m **haga**
Inicio
repetir
seleccionar la siguiente ciudad a visitar con probabilidad P_{ij}^k
hasta hormiga k complete su tour
calcular largo L_k del tour d la hormiga k
fin para
Salvar la mejor solución encontrada hasta el momento
Modifique los niveles τ_{ij} de feromona
Fin

Figure 2: Algoritmo AS

4 Mejoras para Ant System (AS) aplicado a TSP

El objetivo de este estudio es encontrar el conjunto de valores y técnicas que produzcan los mejores resultados para el algoritmo basado en hormigas. Para este efecto se utilizaron diferentes instancias del problema TSP (todas simétricas, y basadas en TSPLIB), y se modificaron los valores de los parámetros pertenecientes al modelo. Además se utilizaron técnicas como "greedy", "hill climbing" y "heurísticas k-opt" para mejorar aspectos específicos del algoritmo.
El algoritmo modificado se muestra en la figura 3.

Inicialmente se utilizaron los valores propuestos por Marco Dorigo [2] en la implementación del algoritmo. Estos valores son:

$$\alpha = 1, \beta = 5, \rho = 0.5, Q = 100, t_0 = 10^{-6}, m = n \quad (5)$$

Para tamaños reducidos se obtuvieron buenos resultados en tiempos razonables, pero en el caso del problema en estudio, los resultados obtenidos fueron, en general, pobres. Las variaciones de los parámetros α y β no entregaron evidencias de mejora, es más, en la mayoría

Procedure AS Hill-Climbing 2-opt 3-opt
Para cada ciclo t desde 1 hasta el número de ciclos **haga**
  Para cada hormiga k desde 1 hasta m **haga**
    Repita
      Seleccionar siguiente ciudad a visitar con probabilidad P_{ij}^k
    Hasta completar su tour
    Calcular largo L_k del tour encontrado
    Si número aleatorio es menor que probabilidad q_0
      Mientras encuentre soluciones mejores
        Aplique 2-opt
      Fin mientras
      Si al aplicar 2-opt no se obtuvo ninguna mejora
        Aplique 3-opt simple
      Fin si
      Actualizar el valor de L_k
    Fin si
  Fin para
  Guardar la mejor solución encontrada hasta el momento
  Actualizar los niveles de fenomona $t_{ij}(t)$
Fin para
Entregar el mínimo tour encontrado

Figure 3: Algoritmo AS hill-Climbing 2-opt 3-opt

de los casos las soluciones obtenidas fueron de peor calidad. En el caso del parámetro m (cantidad de hormigas), la situación fue distinta. La tabla 1, muestra los resultados del algoritmo para diferentes valores de m. En general, se puede observar que para valores de m

m	**promedio**	**mejor**
200	697.336	689.677
100	698.551	697.662
50	700.976	683.683
20	707.632	678.016
10	729.696	727.063

Table 1: Resultados obtenidos con diferentes valores de m para eil100.tsp

grandes, el algoritmo tiende a encontrar buenas soluciones (en promedio). Si bien la cantidad de iteraciones es pequeña (100) y la cantidad de intentos por algoritmo también (5) se puede concluir que valores entre 20 y 100 hormigas entregan los mejores resultados, por tanto se investigará el valor 50 y 20 para el estudio (principio de parsimonia).
Se estudió además la influencia del parámetro t_0 en la solución final obtenida. Para ello, se calculó el valor de t_0 ejecutando un algoritmo greedy para obtener el largo del tour inicial. La matriz de feromona fue inicializada usando el valor $t_0 = \frac{1}{(n*Lg)}$. Adicionalmente, se marcó el tour encontrado con greedy para iniciar la búsqueda. La primera técnica introducida fue heurística 2-opt, con lo cual se mejoraron tanto los promedios como las mejores soluciones encontradas por el algoritmo. La aplicación de esta heurística al finalizar cada generación aumentó la calidad notablemente, debido a que, no solamente se realizó explotación de caminos basada en tour anteriores, sino también a través de un mecanismo externo que pretendía explotar la solución final de cada hormiga, con el fin de obtener mejoras parciales. En este

caso, para cada hormiga de la generación, se envió el tour encontrado para ser mejorado por esta heurística. Los resultados se muestran en la tabla 2. A la luz de los resultados

m	promedio	mejor
50	670.777	660.273
20	674.369	668.617

Table 2: Resultados obtenidos al aplicar heurística 2-opt a cada hormiga

obtenidos, se favoreció la explotación del algoritmo, de forma tal que, aplicase la heurística 2-opt reiteradamente hasta que los valores obtenidos no pudiesen ser mejorados por ella. Entonces, basado en esta idea, fue implementada una variante de algoritmo hill-climbing, tomando como movimiento la heurística 2-opt. Este algoritmo fue aplicado con distintos valores de probabilidad (q_0) para determinar la conveniencia de su aplicación. Los resultados de esta última técnica se resumen en la tabla 3. Si bien los resultados son alentadores,

q_0	promedio	mejor
1	638.929	638.422
0.5	641.588	638.73
0.1	644.054	643.249

Table 3: Resultados obtenidos con diferentes valores de q_0

es necesario destacar que el tiempo de ejecución del algoritmo aumentó considerablemente, por lo que la elección de $q_0 = 1$ para problemas mayores (con un número elevado de ciudades) lo hace impracticable. Por esta razón se consideró como mejor opción mantener una menor probabilidad de exploración vía hill-climbing ($q_0 = 0.5$), e incorporar modificaciones, de tal forma de mantener la performance del algoritmo relativamente estable y mejorar las soluciones obtenidas. Para este efecto, fueron modificados los valores de ρ a 0.3, y el valor de "probabilidad de aplicación de mejoras" $q_0 = 0.5$. Además, se incorporó la heurística 3-opt básica, para todas aquellas soluciones favorecidas con la probabilidad de mejora que no obtuvieron mejoras reales al aplicar hill climbing con 2-opt. El resultado fue un algoritmo equivalente al anterior ($q_0 = 1$) pero con un tiempo de ejecución menor.

Tsplib	valor	q_0	Algoritmo
Ulysses16	73.9876	1	AS 2-opt
Burma14	30.8785	0	AS Simple
St69	677.0510	0.5	AS hill climbing 2-opt
Berlin50	7496.1301	0.5	AS 2-opt
Eil100	637.4480	0.1	AS hill climbing 2-opt
a279	2589.5900	0.1	AS hill climbing 2-opt + 3-opt

Table 4: Resultados de la aplicación del algoritmo para diferentes problemas

5 ACS : Ant Colony System

5.1 Explicación ACS

El algoritmo se muestra en la figura 4 (ver [1] ,[3]). ACS tiene 4 modificaciones con respecto a AS:

Procedure ACS-TSP
inicializar
Para cada arco (i,j) **haga**
$\tau_{ij}(0) = \tau_0$
fin para
Para k=1 **hasta** m **haga**
Ubique la hormiga k en una ciudad elegida aleatoriamente
fin para
Sea T^+ et tour más corto encontrado desde el inicio, y L^+ su largo
Para t=1 **hasta** el número de ciclos **haga**
Inicio
Para k = 1 **hasta** m **haga**
Inicio
repetir
Si existe al menos una ciudad $j \in$ lista de candidatas **entonces**
Elegir la siguiente ciudad $j, j \in J_i^k$, entre las cl ciudades en la lista de candidatas con:

$$j = \begin{cases} argmax_{u \in J_i^k}[\tau_{iu}(t)]^\alpha [\eta_{iu}]^\beta & \text{si } q \leq q_0 \\ J & \text{si } q > q_0 \end{cases} \quad (6)$$

donde $J \in J_i^k$ se elige según la probabilidad:

$$P_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{h \in J_i^k}[\tau_{ih}(t)]^\alpha [\eta_{ih}]^\beta} \quad (7)$$

y donde i es la ciudad actual
Sino elegir la ciudad $j \in J_i^k$ más cercana
fin si
Luego de cada transición la hormiga k modifica localmente:

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t) + \rho\tau_0 \quad (8)$$

hasta hormiga k complete su tour
calcular largo L_k del tour de la hormiga k
fin para
Salvar la mejor solución T^+, L^+, encontrada hasta el momento
Para cada arco $(i,j) \in T^+$ **haga**
Modifique los niveles τ_{ij} de feromona aplicando la siguiente regla:

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t) \quad (9)$$

donde $\Delta\tau_{ij}(t) = \frac{1}{L^+}$
fin para
Fin

Figure 4: Algoritmo ACS: Autores: Dorigo y Gambardella (IDSIA - Suiza). Valores de los parámetros usados en experimentos: $\alpha = 1, \beta = 2, q_0 = 0.9, m = 10, Q = 100, \tau_0 = (nL_{nn})^{-1}, cl = 15$

- Nueva regla de transición,
- Nueva regla de modificación de los trazados de feromona
- Uso de cambios locales de trazados de feromona para favorecer la exploración
- Uso de una lista candidata para restringir la elección de la próxima ciudad a visitar.

Regla de Transición: Se modifica para permitir explícitamente la exploración. Cuando $q > q_0$ la regla de transición es la misma que en AS. Sin embargo, cuando $q \leq q_0$ corresponde a la explotación del conocimiento disponible del problema, es decir, la distancia entre las ciudades y la memoria guardada en los trazos de feromona. Con $q > q_0$ se favorece más la exploración. Cuando q_0 se aproxima al valor 1 se seleccionan sólo soluciones óptimas locales, sin embargo el óptimo local puede no corresponder al óptimo global. Cuando q_0 está cercano a 0 se examinan todas las soluciones locales, aunque tengan mayor peso aquellas óptimas locales (esto difiere de simulated annealing donde todos los estados tienen el mismo peso para ser elegidos a una alta temperatura).

Regla de modificación de los trazados de feromona: En AS todas las hormigas pueden depositar feromona al completar sus tours. En ACS sólo la hormiga que encontró el mejor tour en la iteración puede modificar globalmente las concentraciones de feromona en los arcos. Así en la próxima iteración, las hormigas son motivadas a buscar trayectorias en la vecindad del mejor tour encontrado antes. Otra diferencia es que en AS la modificación del trazado de feromona es sobre todos los arcos, en ACS es sólo sobre aquellos que pertenecen al mejor tour encontrado en la iteración.

Modificación local del trazado de feromona: Cuando la hormiga k está construyendo su tour, se encuentra en la ciudad i y selecciona la ciudad j para continuar, la concentración de feromona del arco (i, j) cambia. Cuando una hormiga visita un arco, la aplicación de la modificación local hace que el nivel de feromona en el arco disminuya. El objetivo es hacer cada vez menos atractivos los arcos visitados por más hormigas, con lo cual se está favoreciendo indirectamente la exploración de los arcos aún no visitados. Una consecuencia de esta estrategia es que las hormigas tienden a no converger a una misma trayectoria o tour.

Uso de la lista candidata: ACS explota una lista de candidatas. Una lista de candidatas es una lista de ciudades preferidas a ser visitadas a partir desde una ciudad dada. En lugar de examinar todas las posibilidades desde la ciudad i, se examinan primero aquellas no visitadas que están en la lista de candidatas y luego las restantes, siempre que no existan ciudades candidatas. La lista de candidatas de una ciudad contiene las cl ciudades más cercanas. Las ciudades están ordenadas de menor a mayor distancia y la búsqueda en la lista es secuencial.

6 Mejoras para Ant Colony System (ACS) aplicado a TSP

6.1 Contaminación

Para controlar el peso que tiene la feromona en el algoritmo, se propone un mecanismo de contaminación, es decir, cada cierto tiempo, la matriz de feromona es contaminada mediante una distribución de probabilidad que tiene el efecto de "remover el suelo"(terremoto). El objetivo es controlar el algoritmo para que no se sesgue con este parámetro y por lo tanto con

Procedure ACS_TSP contaminación aleatorio
Inicialización de Parámetros.
Para (todas las ciudades) crear lista de candidatos de largo *cl*.
Para (Número de Ciclos definido){
  Inicializar matriz de Probabilidades.
  Para (Número de hormigas){
    Para (TOTALCIUDADES){
      Si ($Uniforme() \leq q_0$)
        Seleccionar según $aleatorio(cl)$ de lista candidata que $\in$ permitidos
      sino
        Seleccionar próximo movimiento según ecuación (4)
      Actualizar feromona local
      $q_0 = q_0 * 0.8$
      $\tau = \frac{1}{(TOTALCIUDADES * L_k)}$;
    si el óptimo global cambia{
    si $(q_0 < 0.5)$ $q_0* = 2$;
    }
    si $((Contador > 5 * epoca)$){
    reiniciar q_0;
    para (TOTALCIUDADES)
      para (TOTALCIUDADES){
        si $(i \neq j)$
          $feromona[i][j]* = (\frac{(uniforme() - 0.5)}{4})$;
    }
  Actualizar feromona global
Fin ACS_TSP

Figure 5: Algoritmo ACS, contaminación aleatorio (GRASP)

un camino definido que hará tender al algoritmo a caer en óptimos locales. La contaminación permite que cada cierto número de épocas, las hormigas exploren caminos antes no vistos.

6.2 Parámetro q_0

Se desea controlar el grado de exploración de un algoritmo versus su explotación, adaptando la probabilidad de aceptar el próximo movimiento proveniente de la lista de candidatos o sólo de la lista tabú. El parámetro q_0, decaerá geométricamente durante la ejecución del algoritmo partiendo de 1 y cayendo a 0. El control adaptativo se realiza al detectar que el algoritmo ha quedado atrapado en un óptimo local durante un número de épocas, por lo cual se desea que este explore para que escape y posiblemente encuentre una mejor solución.

Además, de manera de que el algoritmo no sobreexplote una solución cuando encuentra un nuevo óptimo, dependiendo del valor actual, $q_0 < 0.5$ entonces se duplica.

6.3 Lista de Candidatos y elección de próximo movimento basado en GRASP

Cada ciudad posee lista de candidatas que básicamente posee las x ciudades más cercanas o mínimo costo. Es un parámetro que controla el número de movimientos óptimos restringido sólo a la visibilidad, no se considera la feromona, sólo minimizar la distancia del movimiento.

Cuando q_0 tiende a 0 se tiene una alta tasa de explotación, es decir los próximos movimientos a realizarse vendrán primordialmente de la lista de candidatos, se realiza el mejor movimiento

en cuanto al mínimo costo.

El cambio que se propone es, elegir con la técnica GRASP el próximo movimiento perteneciente a la lista candidata (ver [4]) y regular su largo de acuerdo a ella. Las ecuaciones son las siguientes:

$$s = \begin{cases} \text{aleatorio(cl)} & si\ q < q_0\text{(explotación)} \\ \text{escoger probabilísticamente} & \text{sino explorar otras ciudades} \end{cases} \quad (10)$$

Algoritmo 2: ACS, contaminación aleatorio(GRASP).

Una segunda técnica es comparada con ACS, contaminación aleatorio(GRASP), con una modificación que se muestra en la siguiente ecuación:

$$s = \begin{cases} \left\{ \begin{array}{l} \textbf{Si } (uniforme() > 0.5)\text{Ruleta\_candidatos}(cl) \\ \textbf{sino } \{argmax(\tau\eta^{\beta})\} \end{array} \right\} & si\ q < q_0\text{(explotación)} \\ \text{escoger probabilísticamente} & \text{sino explorar otras ciudades} \end{cases} \quad (11)$$

Algoritmo 3: ACS, contaminación quasi GRASP

La técnica que demostró ser superior a las demás intentadas en este trabajo, se describe en la figura 5. Los mejores resultados obtenidos se lograron con el Algoritmo 2, (ACS, contaminacion aleatorio GRASP) los que serán mostrados y analizados en la siguiente sección.

7 Resultados Experimentales para TSP Simétrico

7.1 Objetivos del Experimento

El objetivo de este algoritmo es caracterizar el algoritmo ACS con contaminación aleatorio, de manera individual, es decir analizarlo en el caso del TSP simétrico. Se desea ver cuál es el comportamiento si se trabaja con distintos valores para los parámetros.

7.2 Factores a Explotar

Se tomaron en cuenta los siguientes factores para la experimentación:

- Cl: Número de ciudades pertenecientes a la lista de candidatos.
- δ : Constante geométrica con el cual dinámicamente el parámetro q_0 disminuye.

7.3 Protocolo de Medida de Performance en Problemas TSP

Por cada problema, los datos muestran el mejor resultado obtenido y el promedio obtenido en cinco experimentos, cada uno consistiendo de distinto número de intentos. Cada uno termina después de un número de pasos dada la siguiente fórmula: $n * j$, donde n es el número de ciudades y j varía en un conjunto de $\{1, 5, 10, 50, 100\}$.

7.4 Experimento ACS q_0 variable

Valores fijos: $r = 0.5$, $\alpha = 0.4$, $cl = 20\%$
El número de iteraciones requerido para este algoritmo versus el tiempo que demora, depende del número de ciudades, más bien del conocimiento del problema. Comparado con el siguiente algoritmo, el número de iteraciones no determina que el algoritmo encuentre la solución, sólo se queda más tiempo estancado en un mínimo local (Ver Tabla 5).

Tsplib	valor	δ	Algoritmo
Ulysses16	73.9876	0.975	ACS
Burma14	30.8785	0.975	ACS
St69	677.0510	0.975	ACS q_0
Berlin50	7496.1301	0.975	ACS q_0
Eil100	638.1620	0.975	ACS explorador contaminador
a279	2592.1301	0.975	ACS explorador contaminador

Table 5: Resultados obtenidos al utilizar variaciones del algoritmo con distintas ciudades

7.5 Experimento ACS contaminación aleatorio(GRASP).

Tsplib	cl	valor	δ
St69	33	677.0510	0.9
Berlin50	17	7496.1301	0.975
Eil100	39	637.448	0.9
A279	100	2583.61	0.8

Table 6: Resultados obtenidos

Análisis de Resultados:

El largo de la lista de candidatas *cl* se escogió empíricamente como $\frac{1}{3}$, el rango estudiado fue $[0.25 * n, 0.75 * n]$(Ver tabla 6). El valor de δ afecta directamente la exploración y la explotación. Si se desea una alta exploración, se escoge un valor de δ cercano a 1, pues este parámetro hace caer geométricamente a q_0 hacia el valor 0. Por otro lado, si se desea explotar, entonces se debe escoger un δ cercano a 0. De esta manera se comienza una explotación inmediata, seleccionando movimientos de preferencia pertenecientes a la lista de candidatas.

8 Gráfico de los Tour Iniciales vs. Tour Óptimo

En las Fig. 6 y Fig. 7 se muestran los resultados obtenidos para dos problemas: a279 (279 ciudades) y eil100 (100 ciudades).
En cada una de ellas se ve el tour inicial y el óptimo encontrado por el algoritmo.

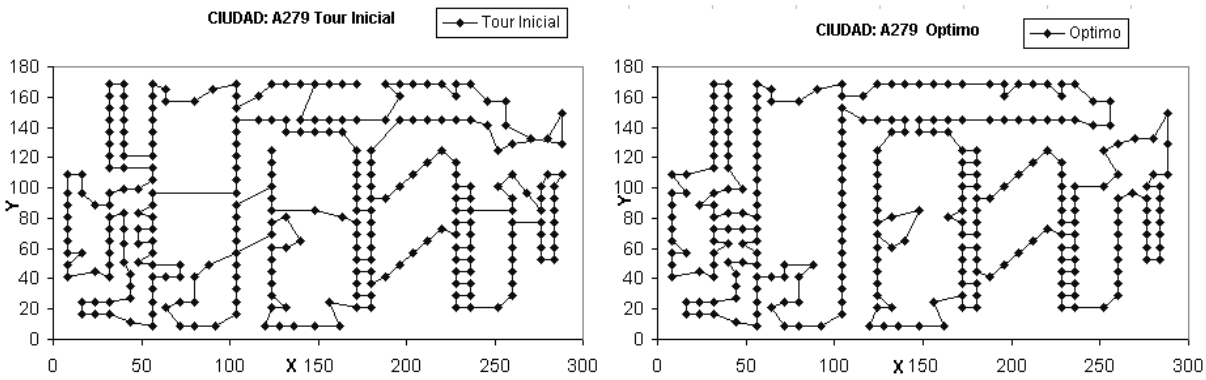

Figure 6: Tour inicial versus Tour óptimo encontrado para un grupo de 279 ciudades

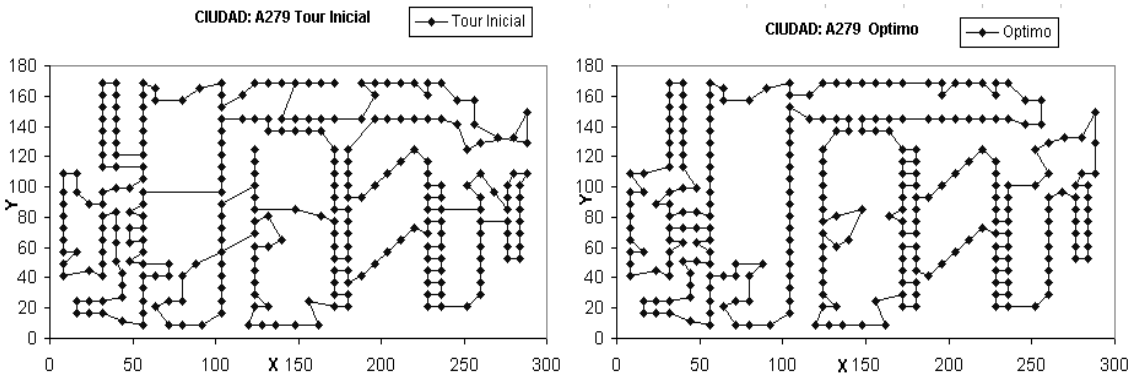

Figure 7: Tour inicial versus Tour óptimo encontrado para un grupo de 100 ciudades

9 Conclusiones

Este paper plantea la combinación de un nuevo paradigma basado en insectos sociales (Sistemas de hormigas) con técnicas bien conocidas de inteligencia artificial para enfrentar problemas de optimización combinatoria, específicamente aplicado al problema clásico del vendedor viajero.
Se constató que es posible mejorar los algoritmos basados en hormigas, combinándolos con variadas técnicas de inteligencia artificial, aumentando su rapidez de convergencia hacia buenos óptimos. Este último punto es de vital importancia, ya que abre un área de estudio con grandes posibilidades de éxito:"La combinación de diferentes técnicas en un algoritmo híbrido para obtener mejores soluciones en problemas de optimización combinatoria".

El Algoritmo ACS contaminación aleatorio, mezcla la aleatoriedad de GRASP en la elección del siguiente movimiento de una lista de candidatos *cl*. Esta aleatoriedad permite al algoritmo independizarse de la feromona, pues escoger el $argmax(feromona * visibilidad^{\beta})$ es equivalente a la selección probabilística de las ciudades que no pertenecen a la lista de candidatas.

El sistema de contaminación de la feromona contribuye al hecho de que se exploren nuevos caminos a partir del actual.

Finalmente, utilizar el rastro de feromona de las hormigas para encontrar un tour óptimo es satisfactorio, sin embargo existe una tendencia natural del algoritmo de quedar atrapado en un óptimo local. Esto requiere del diseño de nuevas estrategias que permitan evadir esta situación, aumentando el grado de exploración.

Las principales contribuciones del artículo pueden ser resumidas en los siguientes puntos:

1. Se mostró las posibilidades de mejoramiento para sistemas basados en hormigas y su combinación con técnicas conocidas de inteligencia artificial para la obtención de soluciones de mejor calidad.

2. Se comprueba que los valores propuestos por Marco Dorigo, para los parámetros $\alpha = 1$ y $\beta = 5$ entregan soluciones de mejor calidad, estableciendo una relación donde la exploración constituye un factor relevante, incluso mayor que la exploración del algoritmo.

3. La performance del algoritmo ACS así como la calidad de las soluciones obtenidas fue mejorada al incluir una estrategia GRASP. Cabe hacer notar que GRASP se mostró sensible al largo de la lista de candidatas, encontrándose como mejor largo un valor de $\frac{n}{3}$.

References

[1] Lawler J.K., Lenstra, A.H.G., and Rinnooy-Kan D.B. The traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1985.

[2] Dorigo M., Maniezzo V., and Coloni A. The ant system optimization by a colony of operatinf agent. *IEEE Transactions on Evolutionary Computation*, 26, 1997.

[3] Riff M.C. Algoritmos basados en hormigas. 2001.

[4] Riff M.C. Metaheurística g.r.a.s.p. 2001.